Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Lovro Žitnik

Priporočilni sistem za slovenske elektronske knjige

diplomsko delo

univerzitetni študijski program
računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik-Šikonja

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Priporočilni sistemi so programska orodja, ki uporabnikom svetujejo pri izbiri koristnih izdelkov ali storitev. Njihova uporaba je zelo razširjena, za svoje delovanje pa uporabljajo podatke o priporočanih objektih, zgodovini uporabnikovih izbir ter tudi o uporabnikih. Pogosto temeljijo na algoritmih strojnega učenja. Izdelajte priporočilni sistem za elektronske knjige v slovenskem jeziku, ki poleg podatkov o uporabnikih in njihovi zgodovini izposoj upošteva tudi jezikovne značilnosti knjig. Te pridobite s pomočjo tehnik obdelave naravnega jezika. Primerjajte uspešnost različnih napovednih metod in jih ovrednotite na zbirki elektronskih knjig, ki so dosegljive v slovenskih knjižnicah. Rešitev kritično analizirajte.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Lovro Žitnik z vpisno številko 63010176 sem avtor diplomskega dela z naslovom:

Priporočilni sistem za slovenske elektronske knjige (angl. *Recommender system for Slovenian electronic books*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika-Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, 5. julija 2016 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Marku Robnik-Šikonji, ki mi je s hitrim odzivom in korektnim odnosom pomagal pri končanju študija ter me usmerjal pri pisanju diplomskega dela.

Hvala tudi sodelavcu Primožu Marnu za vso pomoč in podporo, zavodu Beletrina za zaupanje pri uporabi podatkov, staršem za motivacijo ter nenazadnje življenjski sopotnici Nini Albreht za izkazano potrpežljivost.

— Lovro Žitnik, Ljubljana, julij 2016.

Kazalo

Povzetek

Naslov: Priporočilni sistem za slovenske elektronske knjige

V diplomski nalogi je bil izdelan prototip priporočilnega sistema za izposojo elektronskih knjig. Vir podatkov je bila množica knjig v formatu ePub, anonimiziran seznam uporabnikov ter nabor transakcij med knjigami in uporabniki. Knjigam smo določili številske stilometrične lastnosti, ki smo jih z uporabo ocen SPEC in Laplace razvrstili po pomembnosti. Besedilo smo predhodno razdelili na enote, oblikoslovno označili in lematizirali. Izluščenim vektorjem značilk knjig in uporabnikov smo zmanjšali dimenzionalnost. Za potrebe priporočilnih algoritmov smo knjige in uporabnike razvrstili v skupine. Za klasifikacijo smo uporabili enorazredno metodo podpornih vektorjev ter metodo Elkana and Nota. Osnovali smo več priporočilnih algoritmov s filtriranjem, osnovanim na vsebini, ter s kolaborativnim filtriranjem. Razvite pristope smo testirali na dva načina. Rezultati so pokazali, da je mogoče izdelati priporočilni algoritem na podlagi stilometričnih lastnosti knjig. Boljše rezultate smo dosegli s kolaborativnim filtriranjem.

Ključne besede: priporočilni sistem, e-knjige, procesiranje naravnega jezika, umetna inteligenca, strojno učenje, gručenje, zmanjševanje dimenzionalnosti, enorazredna klasifikacija.

Abstract

Title: Recommender system for Slovenian e-books

The thesis describes a prototype of recommender system for Slovenian e-library. Data includes multitude of books in the ePub format, anonymized list of users and a set of transactions between the books and users. Book content was tokenized, POS-tagged and lemmatized. Data extraction methods were used to extract numerical stylometric features of books. Features were evaluated with SPEC and Laplacian scores. We reduced dimensionality of feature vectors of both books and users and clustered them. We used the method proposed by Elkan and Noto as well as one-class SVM method to classify the books. We constructed several variants of recommender systems based on content and collaborative filtering. The evaluation results show that recommender system using only stylometric features is possible, however, collaborative filtering offers better overall performance.

Keywords: recommender system, e-books, natural language processing, artificial intelligence, machine learning, clustering, dimensionality reduction, one-class classification.

Poglavje 1

Uvod

V slovenskem prostoru ne obstaja platforma za elektronske knjige, kjer bi priporočilni sistem upošteval stil pisanja. Stil pisanja definirajo različne stilometrične lastnosti besedila, ki jih najlažje pridobimo z računalniško obdelavo. Mednje sodijo recimo povprečna dolžina besed v stavku, konkretnost uporabljenega besedišča in odstotek premega govora. Da je stil pisanja relevantna metrika, dokazuje podobnost del istih avtorjev.

Motivacijo za razvoj priporočilnega sistema za področje elektronskih knjig (v nadaljevanju knjig) je gnala radovednost, ali je mogoče izdelati delujoč priporočilni sistem s poudarkom na **stilometričnih lastnostih besedila**.

Priporočilni sistem uporabnikom elektronske knjižnice (v nadaljevanju e-knjižnice) olajša izbiro, katere knjige prebrati. V bistvu gre za **seznam priporočenih knjig**, urejen po relevantnosti in sestavljen za vsakega uporabnika posebej. V množici knjig se uporabnik sam težko odloči, katero si izposoditi. Izbira priporočenih knjig temelji na zgodovini izposoje uporabnika in njemu podobnih uporabnikov.

Priporočilni sistemi so uveljavljeni in uporabnikom poznani predvsem prek spletnih platform, še posebej v okviru spletnih trgovin. Spletnim platformam omogočajo **povečanje koriščenja njihovih storitev**, kot na primer več ogledov posnetkov (primer: YouTube), več rezervacij hotelov (primer: Booking.com) ali povečanje prodaje izdelkov (primer: Amazon). Na

drugi strani uporabnikom **olajšajo odločitveni proces**. Zato so priporočilni sistemi med uporabniki dobro sprejeti oz. celo zaželeni in pričakovani.

Če se omejimo na področje knjig, je najbolj poznan priporočilni sistem spletne knjigarne Amazon, ki prodaja tako fizične kot e-knjige, za določene segmente knjig pa nudi tudi izposojo. Seznam priporočenih knjig za uporabnika se po prijavi nahaja na vstopni strani (z angl. naslovom "More items to consider", v prevodu "Več artiklov v razmislek"). Uporabnikom bolj znana priporočila so na dnu ogledane knjige, kjer je seznam podobnih knjig. V slovenskem prostoru najdemo podoben pristop v spletni knjigarni Emka. Seznam podobnih knjig ponujajo tudi spletne knjigarne založb Cangura, Doria in Didakta. Večina spletnih knjigarn, predvsem manjših, nima priporočil. V splošnem so obstoječi algoritmi za predlaganje enostavni. Predlagajo predvsem knjige istih avtorjev, ko jih zmanjka, pa podobne naslove iz iste kategorije.

Odločitev za uporabo stilometričnih lastnosti knjig je nastala zaradi želje po oblikovanju priporočilnega seznama **stilsko podobno napisanih knjig**. Stilometrične lastnosti številsko opisujejo stil pisanja besedila. Naš priporočilni sistem naj bi tako presegel predloge knjig istih avtorjev ter knjig iz iste kategorije. Uporabljene stilometrične lastnosti so numerične, pridobljene s pomočjo analize besedila in agnostične do avtorja, predhodne kategorizacije (univerzalne decimalne klasifikacije oz. UDK [80]) ter pomena besedila. Kot primer enostavnejših stilometričnih lastnosti bi navedli povprečno število besed v stavku, odstotek uporabljenih pridevnikov v besedilu ter povprečno število zlogov na besedo. Za razliko od klasičnih metod procesiranja naravnega jezika smo poleg vsebinskih v čimvečji meri upoštevali tudi t.i. funkcionalne besede. To so besede oz. besedne enote (sem namreč štejemo tudi ločila), ki same po sebi ne nosijo pomena in pomenske besede povezujejo med seboj. V večini pristopov se te besede, ki so tudi najbolj pogoste, izločijo iz analize s pomočjo seznamov t.i. stop-besed [72]. Nekateri avtorji, kot na primer J. W. Pennebaker [63], zagovarjajo tezo, da lahko samo iz zaimkov (v angl. I, you, they) ugotovimo pomembne lastnosti besedila.

Za analizo besedil smo uporabili metode s področja **procesiranja naravnega jezika** (angl. natural language processing ali skrajšano NLP). Besedilo vsake knjige je bilo treba razdeliti v manjše podenote, torej poglavja, odstavke, stavke in na koncu posamezne besede in ločila. Ta postopek je imenovan **tokenizacija** in predstavlja začetno fazo procesiranja naravnega jezika.

Za oblikoskladenjsko označevanje smo uporabili **strojno učenje**, kjer se program s pomočjo učne množice nauči pravil **označevanja** in jih uporabi na novih primerih. V našem primeru smo program naučili, kako označevati besede v besedne vrste (samostalnik, glagol, pridevnik ...). Kot učno množico smo uporabili **korpus ccGigafida** z že označenimi besedami. Končni rezultat so bile označene besede v besedilih naših knjig.

Besede smo **lematizirali**, kar je postopek določanja osnovne slovarske oblike posameznim besedam. Podoben je krnjenju, le da namesto korena besede pridobimo besedo, ki jo najdemo v slovarju.

Vsaki besedi smo z različnimi metodami izračunali tudi lastnosti, kot so pogostost, število zlogov, konkretnost itd.

Obdelane besede in stavke smo združevali po principu od spodaj navzgor (od besede prek stavka, odstavka in poglavja) v **vektor značilk posamezne knjige**. Gre za zaporedje številskih podatkov, ki opisujejo knjigo, v našem primeru gre za 92 značilk.

Vektor značilk vsaki knjigi določa točko v visokodimenzionalnem prostoru. S pomočjo razdalj ali kotov med vektorji določamo njihovo medsebojno podobnost ter pripadnost gruči.

Iz točk lahko izpeljemo tudi mehanizem za priporočanje knjige. Če si je uporabnik izposodil knjigo A, izberemo njej najbližjo knjigo B in mu jo priporočimo. To je pristop priporočanja, ki ga imenujemo **filtriranje, osnovano na vsebini**.

Uporabniku pa lahko priporočimo tudi knjige, ki so si jih izposodili podobni uporabniki. Ta pristop priporočanja je imenovan **kolaborativno filtriranje**. Podobne uporabnike poiščemo tako, da jih predstavimo kot točke

v prostoru in poiščemo najbližje. Iz njihove zgodovine izposoje potem predlagamo urejen seznam priporočenih knjig.

Z omenjenimi pristopi smo izpeljali več algoritmov, ki so nam vrnili sezname predlaganih knjig. Algoritme smo testirali in ovrednotili s pomočjo metod, predlaganih v literaturi.

Delo vsebuje osem poglavij in tri dodatke. V drugem poglavju predstavimo področje priporočilnih sistemov, njihovo zgodovino ter način delovanja. V tretjem poglavju opišemo, kako smo iz pridobljenih virov izluščili podatke, s tem pridobili vektorje značilk knjig in uporabnikov ter jih shranili v enotno podatkovno bazo. V četrtem poglavju opišemo zmanjševanje dimenzionalnosti, različne metrike in gručenje, ki jih v petem poglavju uporabimo na knjigah in uporabnikih. V šestem poglavju opišemo, kako iz pripravljenih podatkov zasnujemo različne priporočilne algoritme. V sedmem poglavju predstavimo različne načine testiranja priporočilnih algoritmov, opravimo testiranje, predstavimo rezultate in algoritme ovrednotimo. V osmem poglavju zaključimo s pregledom narejenega ter predstavimo ideje za izboljšave. V prilogah imamo tri dodatke, in sicer šifrant podatkov o uporabniku, opis procesiranja naravnega jezika s knjižnico NLTK in opis optimizacije shranjevanja.

Poglavje 2

Priporočilni sistemi

V tem poglavju predstavimo področje priporočilnih sistemov ter povzamemo njihovo zgodovino. Sledi predstavitev načina delovanja, kjer podamo formalno definicijo problema ter predstavimo različne pristope predlaganja artiklov. Na koncu opišemo razliko med iskalniki in priporočilnimi sistemi.

2.1 Predstavitev področja

Priporočilni sistem (angl. recommender system ali recommendation system) so programska orodja ali tehnike za predstavljanje artiklov, koristnih uporabnikom. Artikli so lahko na primer avtomobili, zgoščenke ali računalniki, nepremičnine, turistične zmogljivosti, znamenitosti, skladbe, filmi, novice, povezave, potencialni partnerji itd.

Izbira je povezana s svobodo in avtonomijo. Izbira potrošnih dobrin je z globalizacijo in pojavom spletnih platform postala tako velika, da ljudem namesto občutka svobode daje občutek utesnjenosti. Schwartz moderen odločitveni proces v poplavi artiklov opiše kot tiranijo, ki povzroča bedo. Omeni tudi, da pri odločitvah povzroča paralizo namesto občutka avtonomnosti [51]. To težavo rešujejo kakovostni priporočilni sistemi. Predlagani artikli uporabniku namreč olajšajo odločitveni proces tam, kjer je artiklov ogromno, uporabnik pa ima slab uvid oz. pregled nad njimi.

Razvoj priporočilnega sistema zahteva multidisciplinaren pristop s področij umetne inteligence, interakcije človek–računalnik, informacijske tehnologije, rudarjenja podatkov, statistike, prilagodljivih uporabniških vmesnikov, sistemov za podporo odločitvam, marketinga in psihologije potrošnika.

Priporočilni sistemi so postali samostojno področje raziskovanja v sredini 90. let prejšnjega stoletja. O temi se organizirajo konference [1], znanstvene revije posvečajo posebne številke tej tematiki [3, 29, 31, 83], obravnavajo pa se tudi na fakultetah kot samostojni predmeti [59, 32].

2.2 Zgodovina priporočilnih sistemov

Zgodovino priporočilnih sistemov lahko razdelimo v sledeča obdobja:

Pred letom 1992 so bili narejeni prvi napredki na področju filtriranja vsebin. Gre za prve zametke priporočilnih sistemov.

1992–1998 je obdobje, ko so nastali prvi sistemi za kolaborativno filtriranje (Tapestry podjetja Xerox), prvi priporočilni sistemi, ki uporabljajo oceno uporabnikov (Grouplens) ter prvi sistem za priporočanje filmov (Movielens). Takrat je nastal tudi prvi hibrid med kolaborativnim filtriranjem ter filtriranjem, osnovanim na vsebini, imenovan Fab [24]. Fab je bil priporočilni sistem za priporočanje spletnih strani.

1999–2004 je obdobje, ko je Amazon prijavil patent za kolaborativno filtriranje na podlagi prodajnih artiklov. Leto kasneje je Thomas Hofmann predlagal metodo pLSA s podobno metodologijo, spletni radio Pandora pa je začel s patentiranim projektom glasbenega genoma (Music Genome Project). Njegov konkurent Last.fm je začel izdelovati profile, osnovane na glasbenem okusu poslušalcev.

Obdobje 2005–2009 je najbolj zaznamovala nagrada Netflix (The Netflix Prize), ki je prinesla tudi največji pospešek razvoja področja priporočilnih sistemov. Podjetje Netflix, ki ponuja istoimensko spletno platformo za ogled filmov, je leta 2006 javnosti predstavilo več kot 100 milijonov ocen filmov anonimiziranih uporabnikov in ponudilo nagrado milijon dolarjev tistemu, ki

za 10% izboljša ocene uporabnikov njihovega lastnega algoritma.

To nagrado je leta 2009 osvojilo podjetje BellKor's Pragmatic Cahos team, zmagovalni algoritem pa je hibrid 107 različnih algoritmov, ki se uteženo seštejejo v eno napoved. Ta pristop zagovarjajo kot boljšo alternativo izboljševanju posameznih algoritmov.

Zaradi pravosodnih težav podjetja Netflix so objavo druge nagrade prekinili, potem ko je raziskovalcem uspelo sicer anonimizirane podatke preslikati na drugo podobno platformo in tako ugotoviti identiteto nekaterih uporabnikov [61].

V tem obdobju so osnovne priporočilne sisteme uvedli tudi priljubljeni spletni servisi, kot sta YouTube in Digg, organizirana pa je bila tudi prva samostojna konferenca na temo priporočilnih sistemov.

0d leta 2010 naprej so priporočilni sistemi stalnica naprednejših spletnih platform. Nadgradili so jih z uvedbo kontekstov, tako da so priporočila odvisna od lokacije, zunanje temperature, pomembnih novic ali celo počutja uporabnika, ki ga sistem prebere z obraza. S povečanim raziskovanjem na tem področju imamo na izbiro vedno širši nabor boljših in natančnejših algoritmov, ki rešujejo vedno bolj specifične primere.

2.3 Način delovanja

V tem podpoglavju formalno definiramo problem priporočanja ter predstavimo pristope, ki ga rešujejo.

2.3.1 Formalna definicija problema

Priporočilni sistemi uporabljajo tri vrste **objektov**:

- artikle,
- uporabnike in
- transakcije.

Transakcije so različni tipi bolj ali manj močnih povezav med artikli in uporabniki, recimo ocena ali ogled artikla s strani uporabnika.

Transakcije ločimo na dva tipa [55], in sicer na **eksplicitne** in **implicitne**.

Eksplicitne transakcije so tiste, s katerimi se uporabnik nedvoumno opredeli do artikla, kot na primer:

- ocena artikla od 1 do 5,
- ocena bolj všečnega artikla izmed dveh,
- izgradnja seznama po vrsti urejenih najljubših artiklov.

Implicitne transakcije so tiste, s katerimi se uporabnik posredno opredeli do artikla, in so zahtevnejše za obdelavo:

- obisk strani artikla,
- analiza časa, ki ga je uporabnik preživel na strani artikla,
- nakup artikla,
- analiza afinitete do podobnih artiklov iz drugih virov (recimo Facebook).

V bistvu gre za izračun **uporabnosti** (angl. utility) artikla za posameznega uporabnika na podlagi transakcij. Modeliramo funkcijo uporabnosti $R(u, i)$, kjer kombinacijo uporabnika u (angl. user) in artikla i (angl. item) preslikamo v realno število.

$$i \in Items, u \in Users$$

$$R : Users \times Items \to \mathbb{R}$$

Priporočilo za uporabnike definiramo kot:

$$\forall u \in Users, j = \arg\max_{i \in Items} R(u, i)$$

V resnici si želimo izračunati karseda natančno **napoved uporabnosti** (angl. utility prediction), ki jo označimo z $\hat{R}$. Izračunamo jo za vsako kombinacijo uporabnika z artiklom. Če se osredotočimo na uporabnika, moramo izračunati $\hat{R}(u, i_1), ..., \hat{R}(u, i_N)$ in potem predlagati artikle z najvišjo oceno

uporabnosti $i_{j_1}, ..., i_{j_K} (K \leq N)$. Število predlaganih artiklov K je ponavadi precej manjše od števila vseh artiklov.

Večina priporočilnih sistemov ne izračuna napovedi uporabnosti za vse artikle, ampak se s omeji na omejeno množico artiklov, ki so bližje uporabniku. Kot primer navedimo prikaz odprtih trgovin v bližnji okolici s pomočjo aplikacije Odpiralni časi, ki kot filtrirni kriterij uporabi trenutni čas in fizično lokacijo uporabnika.

2.3.2 Pristopi k priporočanju

Glede na način izračuna urejenega seznama predlaganih artiklov Ricci s sod. [57] ločijo dva pristopa:

- filtriranje, osnovano na vsebini (content-based filtering),
- kolaborativno filtriranje (collaborative filtering).

Filtriranje, osnovano na vsebini

Pri tem pristopu se iz transakcij (ocene, nakupi, izposoje) zgradi seznam želenih lastnosti artiklov posameznega uporabnika, na podlagi katerih se poišče podobne artikle. Primer: če si je uporabnik izposodil knjigo Ivana Tavčarja, mu prikažemo podobno knjigo – torej še kakšno knjigo Ivana Tavčarja.

Kolaborativno filtriranje

Pri drugem pristopu uporabimo pretekle transakcije posameznega uporabnika (ocene, nakupi, izposoje), na podlagi katerih poiščemo podobne uporabnike in predlagamo artikle, s katerimi so imeli relevantne transakcije. Primer: če si je uporabnik A že izposodil knjigo Ivana Tavčarja ter knjigo Bojana Ekselenskega, medtem ko si je uporabnik B izposodil samo knjigo Ivana Tavčarja, mu bo sistem predlagal izbiro podobnega uporabnika A – torej knjigo Bojana Ekselenskega, četudi ji ta ni podobna. Sistem lahko dogradimo s podobnostjo uporabnikov, recimo po starosti, izobrazbi, lokaciji in podobno.

Hibridni pristop

Pristopa lahko kombiniramo, kar se v praksi izkaže kot dobro. Kombinacijo imenujemo hibridni pristop (angl. hybrid approach), končnemu produktu pa rečemo hibridni priporočilni sistem (angl. hybrid recommender system).

2.4 Razlika med iskalnikom in priporočilnim sistemom

Iskalnik je primeren za uporabo takrat, ko uporabnik točno ve, kakšne so njegove potrebe in jih zna opisati s ključnimi besedami. Kot element uporabniškega vmesnika je uveljavljeno vnosno polje, ki mu je dodan gumb za iskanje.

Priporočilni sistem se uporabi, kadar uporabnikove potrebe niso jasne, uporabnik pa nima uvida v množico podatkov. Potreb ni sposoben opisati s ključnimi besedami. Kot element se ponavadi uporablja vertikalen ali horizontalen seznam, ki je mnogokrat opremljen z gumbom za prikaz več predlogov in z drsnikom. Razlike povzema tabela 2.1.

Tabela 2.1: Tabela razlik med iskalnikom in priporočilnim sistemom glede na različne parametre.

	potrebe	**ključne besede**	**element up. vmesnika**
iskalnik	jasne	definirane	vnosno polje
priporočilni sistem	nejasne	nedefinirane	seznam

Meja med pristopoma se vztrajno briše. Zaradi povečane interaktivnosti uporabniških vmesnikov, še posebej tistih na spletu, uporabniki že v vnosnem polju pričakujejo seznam predlogov, povezanih z vpisanim nizom.

Poglavje 3

Luščenje podatkov

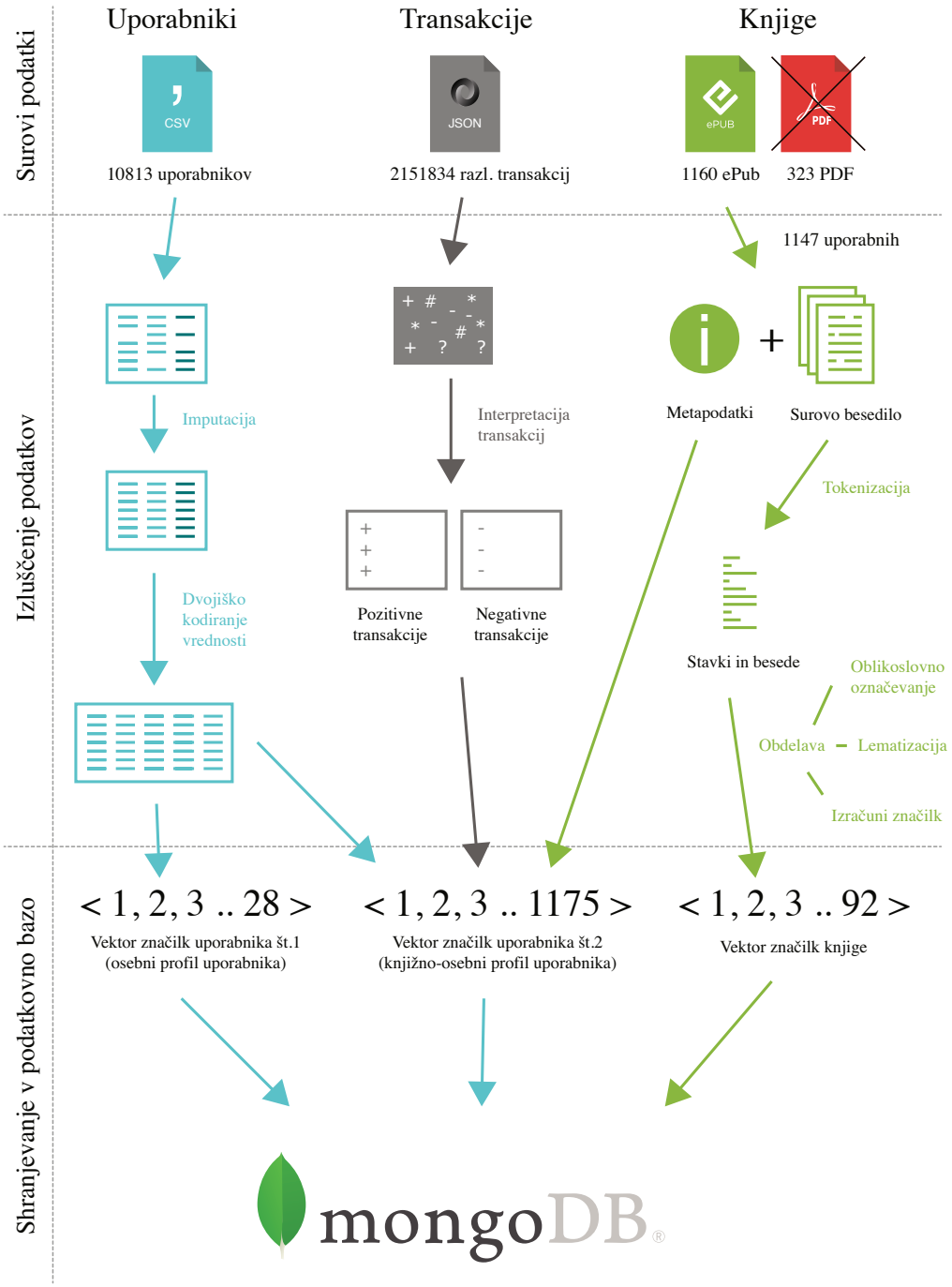
V tem poglavju opišemo, kako smo pridobljene podatke izluščili iz osnovnih besedil, z dodatno obdelavo pridobili številčne vektorje značilk knjig in uporabnikov ter jih shranili v enotno podatkovno bazo. Vektorji značilk knjig in uporabnikov so predpogoj za izračun priporočil.

3.1 Opis postopka

Podatke izluščimo (angl. data extraction) iz nestrukturiranih ali slabše strukturiranih virov podatkov [16]. Namen je nadaljnje procesiranje in/ali shranjevanje izluščenih podatkov.

V našem sistemu smo podatke o knjigah izluščili iz knjig v formatu ePub in izračunali vrednosti različnih atributov. Uporabnikom smo dopolnili manjkajoče podatke, spremenili imenske podatke v številske s postopkom, imenovanim dvojiško kodiranje podatkov. Vsakemu uporabniku smo dodali še vektor značilk s pozitivnimi in negativnimi transakcijami. Vse podatke smo shranili v podatkovno bazo MongoDB.

Postopek izluščenja podatkov ilustrira slika 3.1.



Slika 3.1: Diagram poteka postopka izluščenja podatkov.

3.2 Pridobljeni podatki

Za implementacijo priporočilnega sistema je bilo treba pridobiti dovoljenje za uporabo :

- repozitorija nezaščitenih knjig,
- anonimiziranega seznama uporabnikov in
- zgodovino navad uporabnikov.

Pridobljeni podatki so bili različno strukturirani:

- knjige kot datoteke ePub [23] in PDF,
- seznam uporabnikov v formatu CSV [11] in
- transakcije v formatu JSON [33].

Podatki o knjigah so bili najbolj nestrukturirani. Seznam uporabnikov je bil nepopoln, saj so nekatere vrednosti manjkale. Transakcije so bile omejene v svoji izraznosti, saj ni bilo eksplicitnih transakcij (glej 2.3.1).

3.2.1 Knjige

Na razpolago smo dobili množico datotek, in sicer 323 knjig v formatu PDF in 1160 e-knjig v formatu ePub. Po analizi smo se odločili uporabiti samo knjige v slednjem formatu. Standard PDF je namreč zelo raznolik, v veliki večini pa knjige v tem formatu strukturno niso standardizirane in je izločevanje besedil iz njih zahtevnejše. V mnogih primerih zahteva ročno obdelavo, kar pri tako velikem številu knjig v našem prototipnem sistemu ni bilo mogoče.

3.2.2 Uporabniki

Pridobljena je bila anonimizirana podatkovna baza 10.831 uporabnikov s sledečimi podatki:

- spol,
- leto rojstva,

- stopnja izobrazbe (glej prilogo A.1),
- kategorija po šifrantu COBISS (glej prilogo A.2).

Spol je moški ali ženski. Porazdelitev je neenakomerna, saj je 5.850 uporabnikov ženskega in 2.434 moškega spola (2.529 uporabnikov nima podatka o spolu).

Leto rojstva je dokaj popolno, saj je brez tega podatka samo 291 uporabnikov. Mediana letnice rojstva je 1977, povprečje pa 1980,26.

Stopnja izobrazbe je zadnja stopnja izobrazbe, ki jo je uporabnik dokončal (glej prilogo A.1). Podatki so slabi, saj podatek manjka pri 7.488 uporabnikih, kar je kar 69,13% celotnega vzorca.

Kategorija je življenjski status uporabnika, ki pove, ali se uporabnik šola in na kateri stopnji. Če je že dokončal šolanje, nam pove tip poklica. Če ni več aktivno zaposlen, pa lahko spada med brezposelne ali upokojence (glej prilogo A.2).

Manjkajoče podatke uporabnikov predstavljamo v tabeli 3.1.

Tabela 3.1: Manjkajoči podatki pri uporabnikih.

manjkajoč podatek	**število uporabnikov**	**odstotek uporabnikov**
spol	2.529	23,4%
leto rojstva	291	2,7%
stopnja izobrazbe	7.488	69,2%
kategorija	240	2,2%

3.2.3 Transakcije

Zahtevnost naše naloge je povečala odsotnost eksplicitnih transakcij, torej neposrednih opredelitev uporabnikov do artiklov (glej 2.3.1).

Na voljo so bile samo sledeče implicitne transakcije, ki so se lahko pojavile tudi večkrat:

- obisk strani knjige,

- klik na gumb “izposoja” knjige,
- izposoja knjige,
- predčasna vrnitev knjige.

Obisk strani knjige pomeni, da je uporabnik obiskal stran s podrobnostmi knjige. Obiskov strani iste knjige je lahko tudi več.

Klik na gumb "izposoja"knjige pomeni, da je uporabnik kliknil na gumb izposoja, ni pa nujno tudi dokončal izposoje. Gumb izposoja je na voljo na seznamih najbolj izposojanih knjig na prvi strani, na seznamih v kategorij ter na strani s podrobnostmi knjige.

Izposoja knjige pomeni, da je uporabnik kliknil na gumb "izposoja"knjige in si jo tudi izposodil. Podatek je pomemben, ker ima uporabnik v določenem časovnem preseku lahko izposojene samo štiri knjige. Izposoja brez predčasne vrnitve traja 14 dni.

Predčasna vrnitev knjige pomeni, da je uporabnik že izposojeno knjigo vrnil predčasno, torej v manj kot štirinajstih dneh od izposoje. S tem si je sprostil mesto za ponovno izposojo.

Za vsako transakcijo smo imeli tudi **časovni žig** (angl. timestamp), torej datum in čas, kdaj je do transakcije prišlo. Podatek je pomemben zato, da lahko pripravimo po času urejen seznam izposoj za simulacijo uporabnikovega obnašanja (glej 7.2.1).

Interpretacija transakcij

Izluščiti je bilo treba pomen zgoraj naštetih transakcij. Ljudje smo nezanesljivi pri pripisovanju pomembnosti dogodkom, zato smo se želeli izogniti problemu kognitivne pristranskosti (angl. cognitive bias) [78] pri ocenjevanju, koliko je recimo izposoja knjige pomembnejša od obiska strani knjige (ali je dvakrat, petkrat ali desetkrat pomembnejša).

Domnevali smo, da je edina res **pozitivna transakcija** samo **izposoja knjige**. Da si uporabnik knjigo izposodi, je treba namreč sprejeti odločitev in opraviti obe predhodni transakciji, torej obisk strani knjige ter klik na gumb “izposoja” ter nato tudi potrditev izposoje. Hkrati je uporabnik omejen na

štiri hkratne izposoje knjig za obdobje 14 dni. To pomeni, da odločitev za izposojo s sabo nosi posledice oz. omejitve pri prihodnjih izposojah. Te transakcije smo združili v časovno urejen seznam t.i. pozitivnih knjig, s tem da smo od večkratnih izposoj iste knjige upoštevali samo prvo.

Odločili smo se, da **klike na gumb "izposoja"** zanemarimo, saj podatek ni zanesljiv. Uporabnik je morda samo preveril, ali si knjigo lahko izposodi, uporabniški vmesnik o tem predhodno namreč ne nudi nobene informacije. Omejitev pri izposoji je več. Uporabnik ima lahko hkrati izposojene do štiri knjige, knjižnice knjige nimajo zakupljene ali pa je dosežena maksimalna količina hkratnih izposoj te knjige. Razlogi za to so finančne narave.

Tudi uporabo **predčasne vrnitve knjige** smo zaradi dvoumne interpretacije izpustili. Uporabnik je lahko knjigo vrnil, ker mu ni bila všeč, ali ker jo je zaradi zanimivosti hitro prebral.

O **negativni transakciji** smo razmišljali o enkratnem obisku strani brez drugih transakcij v povezavi s to knjigo. Domneva temelji na razmisleku, da če uporabnik strani knjige ni obiskal večkrat, niti ni kliknil na gumb izposodi ali si jo izposodil, ga knjiga ni dovolj zanimala. Te transakcije smo izluščili in združili v po času obiska urejen seznam t.i. negativnih knjig.

3.3 Enotna podatkovna baza

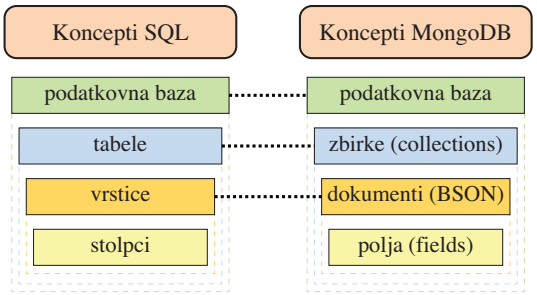
Podatke vseh treh objektov (knjige, uporabniki, transakcije) smo shranili v enotno podatkovno bazo MongoDB. Gre za brezplačno, odprtokodno večplatformsko dokumentno podatkovno bazo. MongoDB spada med NoSQL podatkovne baze, kar pomeni, da ni klasičnih relacij med tabelami, shema pa je dinamična. Ravno ta fleksibilnost dinamične sheme podatkovne baze je bila glavni odločitveni dejavnik, saj prototipni razvoj zahteva nenehne spremembe podatkovne baze.

Dodatni odločitveni dejavniki so bili:

- delovanje na operacijskem sistemu OSX,
- knjižnica za python (PyMongo),

- gonilniki in knjižnica za PHP (PHPLIB),
- kakovostno orodje za administracijo – program MongoChef (3T Software Labs).

Glavni koncepti v MongoDB so analogni podatkovnim bazam SQL, z razliko, da je posamezna vrstica oz. dokument in s tem tudi posamezno polje (field) kar cel objekt, zapisan v formatu BSON [6]. Format BSON je izpeljanka formata JSON (*binary JSON*). Analognost konceptov je prikazana na sliki 3.2



Slika 3.2: Analognost konceptov SQL in MongoDB.

3.4 Uporabniki in transakcije

Pri uporabnikih **podatki ponekod manjkajo** (ni določenega spola, ni letnice rojstva ipd.). Te vrednosti je treba dopolniti, saj implementacija algoritma, ki smo jo izbrali, deluje le na popolnih podatkih. Dodaten izziv je, da **podatki o uporabnikih niso samo številčni**, temveč kategorični:

- podatek o spolu je binaren nominalen,
- podatek o kategoriji je nominalen,
- podatek o izobrazbi je ordinalen.

Poleg tega je treba v vektor značilk uporabnika dodati pozitivne in negativne transakcije. Rešitve za te izzive predstavljamo spodaj.

3.4.1 Imputacija

Imputacija (angl. imputation) je postopek, s katerim dopolnimo manjkajoče vrednosti spremenljivk. Najenostavnejša je dopolnitev z najpogostejšo vrednostjo spremenljivke. Če recimo manjka podatek o spolu in je v vzorcu več moških kot žensk, izberemo moški spol. Ta pristop ne upošteva ostalih podatkov o uporabniku. Če imamo na voljo tudi starost, je na primer za starejše uporabnike verjetneje, da so ženskega spola, saj ženske v povprečju dosegajo višjo starost.

Takšne podatke upoštevajo metode, ki uporabljajo nadzorovano strojno učenje (torej klasifikator ali kombinacijo več klasifikatorjev), za to, da se iz obstoječih podatkov naučijo predvideti manjkajoče podatke. Uporabili smo t.i. "Model-based imputer", ki je del orodja Orange [49]. Knjižnice *scikit-learn* nismo uporabili, saj ima implementirano le implementacijo s povprečjem, mediano in modusom, deluje pa le na številskih spremenljivkah [68].

Končen rezultat so pridobljene manjkajoče vrednosti, ki smo jih shranili v ločen stolpec, imenovan *imputed_fv*.

3.4.2 Dvojiško kodiranje vrednosti

Dvojiško kodiranje vrednosti (znano tudi kot vektorizacija slovarja) [47, 48, 67] je postopek, ki rešuje težavo imenskih atributov tako, da različne vrednosti, ki jih atribut lahko zavzame, dodamo kot stolpce (značilke). Vrednosti teh novih značilk nastavimo na 0, razen v stolpcu, ki je enak vrednosti značilke, kjer je vrednost 1.

3.4.3 Končna vektorja značilk uporabnika

Končna vektorja značilk uporabnika, ki ju uporabimo v sistemu, se razlikujeta v tem, da ne vsebujeta podatkov o transakcijah (osebni profil) ali pa jih (knjižno-osebni profil). Oba kasneje uporabimo za iskanje podobnih uporabnikov (glej 6.2.4).

Osebni profil uporabnika S postopkom dvojiškega kodiranja vrednosti smo iz kategorije izpeljali 25 novih značilk, po eno za vsako možno kategorijo. Pri spolu smo izpeljali eno samo značilko z vrednostmi 0 za moški spol in 1 za ženski spol. Dodali smo obstoječe letnice rojstva in stopnje izobrazbe. Končen vektor značilk ima torej 28 dimenzij in je shranjen v stolpcu, imenovanem *fv_profile*.

Knjižno-osebni profil uporabnika V dodaten vektor značilk, imenovan *fv_profile_books*, smo vnesli še transakcije, saj je ta podatek pomemben za gručenje. Dodali smo 1.147 značilk (stolpcev) za vsako knjigo, označenih s kodo ISBN posamezne knjige. Za pozitivno transakcijo (izposoja knjige) smo vnesli vrednost 1, za negativno pa –1 (enkraten obisk knjige). Če o uporabnikovem odnosu do knjige nismo imeli podatka, smo vnesli vrednost 0. Dodali smo še vse značilke iz osebnega profila uporabnika. Končna dimenzionalnost tega vektorja značilk je torej 1.175.

3.5 Izluščenje vsebine knjig iz formata ePub

Format ePub

Gre za razširjen in uveljavljen format e-knjig s končnico .epub, ki ima široko podporo za prikaz na različnih napravah. Večina trenutno dostopnih knjig je na voljo v standardu 2.0 iz leta 2007. Organizacija IDPF (International Digital Publishing Forum) [30] je leta 2014 izdala specifikacije različice 3.0. Vsebina knjige je zbrana v eni datoteki, ki je zbirka stisnjenih datotek v arhivskem formatu ZIP [17]. V njej najdemo poglavja kot HTML datoteke, ki so jim dodane datoteke s stilom prikaza. Dodane so še datoteke z metapodatki, datoteka s tipom MIME in datoteka s kazalom.

Glavne lastnosti formata ePub so [23]:

- tekoča vsebina, kar pomeni, da je razporeditev vsebin po zaslonu prepuščena prikazovalniku (glede na velikost črk);
- podpora metapodatkom, ki pa niso obvezni;

- podpora rastrskim in vektorskim grafičnim formatom datotek;
- možnost vključitve lastne tipografije in stilov CSS;
- možnost spremembe stilov in tipografije (pomembno predvsem za osebe s težavami z vidom);
- zaznamki.

Največja **pomanjkljivost** z našega stališča je neobveznost izpolnjevanja osnovnih metapodatkov, zaradi česar jih založbe rade izpustijo. Slabo je realizirana tudi urejenost besedila v poglavja, saj format ne loči vsebine knjige od predgovora, predstavitve avtorja, strani s kataložnim zapisom CIP in podobno. Format tudi ne ureja razdelitve knjig v več delov, če jih knjiga ima.

Knjižnica ebooklib

Potrebovali smo knjižnico, ki nam iz datotek ePub izlušči strukturirane podatke. Po testiranju več knjižnic smo se odločili za knjižnico ebooklib [22].

Primer enostavne uporabe:

```
1 import ebooklib
2 book = epub.read_epub("knjiga_2331.epub") # datoteka ePub na disku
3 title = book.title # naslov knjige
4 uid = book.uid  # koda ISBN
```

Knjižnica ni pravilno prebrala imen poglavij, ki so imela v naslovu datoteke presledek, zato smo izločili še dodatnih 13 knjig, tako da je bilo na koncu shranjenih in uporabljenih 1147 knjig.

Pomanjkljivost metapodatkov Neposredno uporabni metapodatki v knjigah so bili pomanjkljivi, kar prikazuje tabela 3.2.

Najslabše smo izluščili najpomembnejši metapodatek, imenovan UID, kjer smo pričakovali kodo ISBN. Ta je bila zamišljena kot naš **unikatni identifikator knjige**, ki smo ga nameravali uporabljati v podatkovni bazi za povezovanje knjig s transakcijami in uporabniki.

Ta metapodatek je v resnici vseboval:

- spletni naslov do nakupa knjige,

Tabela 3.2: Pomanjkljivosti metapodatkov v knjigah (format ePub).

problem	**št. e-knjig**
ni ustreznega ISBN	534
ni naslova	133
ni avtorja	276
ni založnika	253
jezik – angleščina (privzet)	256
ni datuma izdaje	145

- avtomatsko generirane indekse v različnih formatih, ki jih avtomatsko dodajo programi za izdelavo ePub datotek ali
- vrednost "null".

Če je koda ISBN obstajala, je bila zapisana dokaj raznoliko, na primer:

- URN:ISBN:978-9-6120-3366-8,
- 978-961-204-549-4,
- Urn:isbn:9789612303778,
- ISBN 978-961-254-788-2,
- ISBN ???? 978-961-254-796-7.

54 datotek ePub je imelo definirane napačne kode ISBN. Sklepamo, da so založbe uporabile vzorec druge e-knjige in kode ISBN niso popravile.

Zaradi nezanesljivosti kode ISBN, ki je bila nujna kot unikatni identifikator, smo kodo ISBN naknadno sinhronizirali z izvorno podatkovno bazo prek imen datotek. Na isti način smo prenesli tudi metapodatke o avtorju, založbi in letnici izdaje. Podatki v izvorni podatkovni bazi so namreč zanesljivejši.

Vsebina knjig v datoteki ePub

Vsebina knjig v datoteki ePub je po poglavjih razdeljena v datoteke. Če je knjiga sestavljena iz več delov, ni enotnega pravila zapisa. Nekatere knjige imajo to rešeno z gnezdenjem poglavij v podpoglavja v kazalu, kjer poglavja

potem predstavljajo dele knjige. Nekatere knjige nimajo gnezdenja, ampak vizualno zamikanje podpoglavij v kazalu. Vizualno zamikanje je težko programsko razbrati iz kode. Ker je teh knjig malo, smo uporabili samo možnosti, ki jih ponuja knjižnica ebooklib ter v podatkovno bazo prenesli posamezna poglavja.

Knjižnica BeautifulSoup

Raznolikost v HTML formatiranju posameznih poglavij je bila precejšnja, na primer:

- za presledek je bilo ponekod uporabljeno zaporedje znakov "&*nbsp*;";
- kot presledek med odstavki je bil ponekod dodan prazen odstavek "< *p* > &*nbsp*; < /*p* >", kljub dejstvu, da je to mogoče definirati s stilom CSS;
- težave s šumniki.

Telo (angl. body) HTML datoteke poglavja nam je izluščila knjižnica ebooklib, za ter za kodiranje v UTF-8 format pa smo uporabili knjižnico BeautifulSoup. Primer uporabe:

```
1 from bs4 import BeautifulSoup
2 def remove_html(in_string):
3   soup = BeautifulSoup(in_string, 'html.parser') # removes nicely also "<p> </p>"
4   return soup.get_text() # if (strip=True) => performs also strip
```

Prenos besedila v MongoDB

S kombinacijo knjižnic ebooklib in BeautifulSoup nam je uspelo v MongoDB prenesti posamezna poglavja knjig. Besedila poglavij smo shranili v obliki navadnega besedila (angl. plain-text). Pri tem smo se zaradi pohitritev poslužili shranjevanja s svežnjem operacij, ki je opisan v dodatku C.

3.6 Pridobitev značilk knjige

V podpoglavju opišemo, kako smo iz besedila pridobili stilometrične lastnosti knjige. Končni rezultat je vektor značilk knjige, kot je prikazan v tabeli 3.3

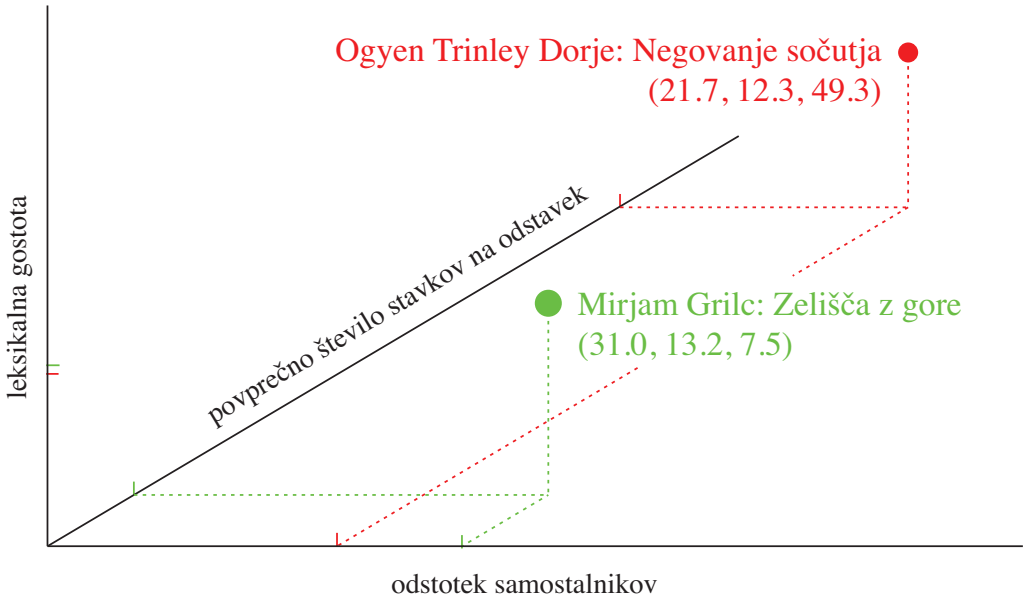
Tabela 3.3: Primer tabele, v kateri je shranjen vektor značilk knjige.

	1	2	3	4	...	92
knjiga \značilka	št. stavkov s premim govorom	odstotek lastnih besed v besedilu	raznolikost lem	število vejic	...	povprečno število zlogov na stavek
Foucaultovo nihalo, Umberto Eco	1.144	2,1689712	7,3448578	22.686	...	3,8153968

3.6.1 Opis postopka

S prenosom besedila poglavij v podatkovno bazo lahko začnemo z analizo knjig. Treba je izračunati stilometrične lastnosti besedila, ki tvorijo vektor značilk knjige, torej urejen seznam lastnosti knjige. Posamezno knjigo si tako lahko predstavljamo kot **točko v 92-dimenzionalnem prostoru**, kjer posamezne dimenzije predstavljajo lastnosti knjige.

Za ilustracijo vzemimo primer dveh knjig s tremi značilkami. Tako dobimo dve točki v tridimenzionalnem prostoru, kar ponazarja slika 3.3. Uporabljeni podatki so v tabeli 3.4.



Slika 3.3: Prikaz dveh knjig kot točk v treh dimenzijah.

Do značilk pridemo tako, da najprej razdelimo besedilo na podenote, torej

Tabela 3.4: Tabela dveh knjig s tremi izbranimi značilkami.

knjiga / značilka	odstotek samostalnikov	leksikalna gostota	povprečno število stavkov na odstavek
Ogyen Trinley Dorje: Negovanje sočutja	21,7	12,3	49,3
Mirjam Grilc: Zelišča z gore	31,0	13,2	7,5

na poglavja, odstavke, stavke in besede oz. besedne enote. Temu postopku pravimo **tokenizacija** in je obrazložena v dodatku B.1. Nato za besede in stavke izračunamo osnovne atribute in iz njih izračunamo značilke knjige. Pri tem si pomagamo s postopki **procesiranja naravnega jezika** (angl. natural language processing), ki so opisani v dodatku B.

Postopek je sledeč: vsaki besedi oz. besedni enoti pripišemo njene osnovne atribute, kot so: lema, oblika v malih črkah (angl. lower-case), število zlogov in besedna vrsta. Besedno vrsto dobimo s postopkom **oblikoslovnega označevanja besedila** (glej dodatek B.2), ki uporablja tudi **lematizacijo** (glej B.3), s katero iz besede pridobimo osnovno slovarsko obliko, imenovano lema. Postopek je podoben krnjenju.

Nato izračunamo **značilke, vezanih na besedo**. Na primer število zlogov na besedo izračunamo s seštevkom števila zlogov vseh besed in delitvijo s številom besed.

Sledi združevanje atributov v stavku ter **izračun značilk, vezanih na stavek**. Tako na primer izračunamo odstotek premega govora v stavku.

S podobnim postopkom nadaljujemo do nivoja knjige in izračunamo na primer značilko število stavkov s premim govorom, kjer preštejemo stavke s pozitivnim odstotkom premega govora.

Zaradi uporabe algoritmov, ki delujejo le na numeričnih podatkih, smo se pri knjigah izračunali samo numerične atribute. Vnos imenskih atributov namreč prinaša omejitve pri uporabi algoritmov.

3.6.2 Opis značilk knjige

V tem razdelku predstavimo pomen posameznih značilk knjig in kako smo do njih prišli.

Pogostost besed

Zbirko besed smo izluščili iz korpusa ccGigafida [60]. Sprehodili smo se skozi celoten korpus in v podatkovno bazo MongoDB dodajali neobstoječe besede, že obstoječim pa povečali štetje. Čas celotnega procesiranja je znašal 28 ur, 49 minut in 23 sekund. Besede smo pustili v izvirni obliki, torej jih nismo spreminjali v male črke ter tako ohranili velike začetnice. Na koncu smo besede razvrstili padajoče po številu pojavitev v korpusu, vrstni red vnesli kot polje v MongoDB in za pohitritev to polje indeksirali. Iz vrstnega reda smo za posamezne besede pridobili frekvenco na skali od 0 do 10. Zaradi pogostosti najpogostejših besed je bilo treba ločiti zgornji del po desetiški logaritemski skali, in sicer prvih 0,001 % ima stopnjo pogostosti 10, 0,001 % - 0,01 % stopnjo pogostosti 9 in tako dalje.

Pogostost lem

Za pogostost lem smo uporabili isti postopek kot pri pogostosti besed, le da smo besedo prej pretvorili v male črke in jo nato lematizirali. Čas procesiranja celotnega korpusa je bil 28 ur, 36 minut in 15 sekund.

Konkretnost besede in starost usvojitve besede

Za slovenski jezik nismo našli metode, ki bi besedi določila **konkretnost** (ali obratno – abstraktnost). Kot najboljši približek smo uporabili seznam besed v angleščini z definirano konkretnostjo [12]. Ker so besede simboli za koncept, se nam prevajanje ne zdi problematično.

Tudi za **starost usvojitve besede** nismo našli slovenskega seznama in smo uporabili angleški seznam [2]. Prevajanje angleških besed v slovenske ekvivalente je tu težavnejše, saj govorimo o besedi in ne konceptu, ki ga beseda simbolizira. Naj ponazorimo na primeru tujk: pri estimaciji in oceni gre za isti koncept. Ponavadi besedo ocena usvojimo precej mlajši kot besedo estimacija. Strojni prevajalnik pri tem ne dela razlik in izbere najenostavnejšo slovensko besedo.

Tabela 3.5: Povprečne vrednosti izračunanih indeksov za knjige, združenih po kategorijah.

kategorija	vel. vzorca	Gunningov indeks	enostavnost branja po Fleshu	ocena nivoja Flesh-Kincaird
e-knjige za otroke	81	7,26	39,76	10,47
e-knjige za mladostnike	25	8,41	33,63	11,68
ljubezenski romani	69	7,91	35,68	11,15
poezija in dramatika	161	8,22	34,21	11,80
naravoslovje, tehnika, matematika	5	12,04	10,88	15,93
humanistika in družboslovje	96	13,17	10,13	16,63

Metodologija prevajanja besed Urejen seznam za konkretnost in starost usvojitve besede v angleščini (vrednosti, ločene z vejico) smo povzeli iz repozitorija Quijada s sod. [86]. S pomočjo python knjižnice TextBlob smo prevedli tabelo. Neznanih prevodov v analizi nismo upoštevali. V knjižnici TexBlob neznani prevod zaznamo tako, da je prevod isti kot vhodna beseda. Knjižnica deluje prek storitve Google Translate in ima omejitev, koliko zahtev lahko sprocesira v določenem času. Besede smo združevali v večje skupine besed (optimalna izbira je bila 200, ločili smo jih z zaporedjem " | "), pognali prevajanje in rezultat zopet razdružili, saj so vrstni red in ločitveno zaporedje ostali enaki. Vmes smo uporabili python funkcijo za zakasnitev, s katero smo počakali 5 s med dvema zahtevkoma (*time.sleep(5)*).

Primer enostavne uporabe knjižnice TextBlob:

```
from textblob import TextBlob
blob = TextBlob(blob_of_words_en)
blob_of_words_sl = blob.translate(from_lang="en", to='sl')
```

Indeksi zahtevnosti besedila

Gre za indekse, katerih parametri so umerjeni po značilnostih angleškega jezika in definirajo zahtevnost besedila [84]. Porajalo se je vprašanje njihove uporabnosti za slovenščino, vendar je pregled literature pokazal, da jih v slovenskem jeziku uporabljajo tudi drugi avtorji [79, 5]. Indeks se uporablja tudi v slovenski različici orodja za pisanje besedil Microsoft Word [54]. Uporabnost dokazuje tudi tabela 3.5. V njej smo izračunali povprečne vrednosti indeksov za knjige, združene po kategorijah. Iz kategorije lahko sklepamo na zahtevnosti, ki se odražajo na indeksih.

Ocena enostavnosti branja po Fleschu Metoda (angl. Flesch reading-ease score) [26] ocenjuje zahtevnost branja teksta za bralca. Ker uporablja parametre, ki jih že imamo izračunane, smo to oceno vključili v vektor značilk knjige. Veljavne ocene so od 0 do 100. Enačba za izračun je:

$$206,835 - 1,015 \cdot \left(\frac{\check{s}t.\ vseh\ besed}{\check{s}t.\ vseh\ stavkov} \right) - 84,6 \cdot \left(\frac{\check{s}t.\ vseh\ zlogov}{\check{s}t.\ vseh\ besed} \right)$$

Ocena nivoja Flesh-Kincaid Ocena nivoja Flesh-Kincaid (angl. Flesh-Kincaid grade level) [27] nam pove, za kateri razred ameriškega izobraževalnega sistema je tekst primeren. V redkih primerih se lahko zgodi, da dobimo negativen rezultat. Enačba za izračun je:

$$0,39 \cdot \left(\frac{\check{s}t.\ vseh\ besed}{\check{s}t.\ vseh\ stavkov} \right) + 11,8 \cdot \left(\frac{\check{s}t.\ vseh\ zlogov}{\check{s}t.\ vseh\ besed} \right) - 15,59$$

Gunningov indeks Gunningov indeks (angl. Gunning fog index) [28] nam pokaže, na katerem nivoju ameriškega izobraževalnega sistema tekst ne bi smel povzročati težav z razumevanjem. Tudi ta indeks je umerjen na angleški jezik, vendar smo ga vseeno vključili. Enačba za izračun je:

$$0,4 \cdot \left(\frac{\check{s}t.\ vseh\ besed}{\check{s}t.\ vseh\ stavkov} \right) + 100 \cdot \left(\frac{\check{s}t.\ kompleksnih\ besed}{\check{s}t.\ vseh\ besed} \right)$$

Kompleksna beseda je beseda, ki ima tri zloge ali več. Kot zlogov ne smemo upoštevati pogostih končnic besed (-ji, -ti, -anje ipd.). Iz kompleksnih besed je treba izločiti lastna imena, vsakdanji žargon in sestavljene besede.

Preostale pomembnejše značilke

Večino preostalih značilk smo povzeli po Fengu s sod. [10], nekaj pa smo jih zasnovali sami.

Leksikalna gostota je mera za gostoto vsebinskih besed [39]. Definirana je kot:

$$100 \cdot \left(\frac{\check{s}t.\ vsebinskih\ besed}{\check{s}t.\ vseh\ besed} \right)$$

Pod vsebinske besede spadajo samostalniki, pridevniki, glagoli ali prislovi. Preostale besede poimenujemo funkcionalne, saj imajo funkcijo povezovanja vsebinskih besed.

Raznolikost lem pove delež različnih lem glede na vse besede. Definirali smo jo sami, in sicer:

$$100 \cdot \left(\frac{\check{s}t.\ razli\check{c}nih\ lem}{\check{s}t.\ vseh\ besed} \right)$$

Raznolikost besed je delež različnih besed glede na vse besede. Definirali smo kot:

$$100 \cdot \left(\frac{\check{s}t.\ razli\check{c}nih\ besed\ v\ malih\ \check{c}rkah}{\check{s}t.\ vseh\ besed} \right)$$

Značilke s premim govorom in citati pridobimo s štetjem besed, ki nastopajo v premem govoru ali v citatih. Po testiranju nekaterih knjižnic smo zaradi njihove nezanesljivosti napisali lastno, ki upošteva različne tipe narekovajev in prepozna citat na začetku stavka.

Značilke s stop-besedami uporabljajo koncept stop-besed (angl. stop-words), ki je nastal zaradi hipoteze, da so določene besede prepogoste, da bi jih lahko upoštevali kot pomembne pri procesiranju naravnega jezika. Pogosto jih izločimo glede na namen obdelave. Seznami stop-besed vsebujejo predvsem veznike, številke, medmete, skratka besede, ki spadajo v skupino t.i. funkcionalnih besed. Uporabili smo seznam, v katerem je 449 besed [73].

3.6.3 Končen seznam značilk knjig

Končen seznam vseh 92 značilk, ki opisujejo vsako obdelano knjigo, je v tabeli 3.6. V tabeli je tudi opis značilke ter razvrstitev po pomembnosti z metodama SPEC in Laplace.

Ocena Laplace je namenjena izbiri značilk pri neoznačenih podatkih. Izbere značilke, ki najbolje ohranjajo strukturo podatkov. Sestavljena je iz treh faz. Prva je konstrukcija grafa najbližjih sosedov (angl. nearest neighbor graph) med točkami, iz katere izračunamo podobnostno matriko povezanih točk s posebno metriko. Iz obeh matrik izračunamo Laplacovo matriko, nato pa za vsako značilko posebej izračunamo oceno Laplace (angl. Laplacian score). Nižja kot je ocena, pomembnejša je značilka. Podrobnosti postopka opisuje He s sod. [35].

SPEC je algoritem, ki tako označene kot neoznačene podatke oceni in razvrsti po pomembnosti na podlagi spektralne analize grafov [71]. Algoritem najprej izračuna podobnostno matriko (angl. similarity matrix) in iz nje skonstruira neusmerjen graf podobnosti. Nato se sprehodi po podatkih in za vsako značilko primerja, kako konsistentna je pri pripisovanju podobnih vrednosti instancam, ki so si v grafu bližje. Značilkam, ki so pri priposovanju konsistentnejše, določi višjo pomembnost. Pri tem uporablja tri ocenjevalne funkcije. Postopek podrobno opisujeta Zhao in Liu [70].

Analiza ocen Laplace in SPEC nam pokaže, da obe metodi podobno ocenjujeta pomembnost značilk. Kot najpomembnejšo značilko predlagata raznolikost besed. Visoko se pojavlja tudi delež zaimkov (3. po oceni Laplace in 5. po SPEC), s čimer bi lahko potrdili domnevo avtorja knjige Skrivnostno življenje zaimkov J. W. Pennebakerja [63], da so zaimki pomemben razločevalni element besedil kljub navidezni nepomembnosti. Zaimki namreč spadajo med funkcijske besede in so marsikje uvrščeni na seznam stop-besed, ki so iz analiz izvzete. Najmanj pomembne so značilke "število stavkov s šestimi vejicami" (po oceni Laplace) in "število stavkov s trinajstimi vejicami" (po SPEC). Metodi se najbolj razlikujeta v razvrščanju značilke "število zlogov v stavku", saj ji ocena Laplace prisodi 91., torej predzadnje mesto, SPEC pa 48. mesto. Drugače razvrstita tudi "število stavkov s premim govorom"; ocena Laplace ji prisodi 46., medtem ko ji SPEC prisodi 91. mesto. Drugje so razlike manjše.

Tabela 3.6: Seznam vseh izračunanih značilk knjige, kjer stolpca SPEC in Laplace predstavljata razvrstitev pomembnosti značilke.

št.	**oznaka v podatkovni bazi**	**opis značilke**	**Laplace**	**SPEC**
1	word_diversity	raznolikost besed	1	1
2	lemma_diversity	raznolikost lem	65	65
3	lexical_density	leksikalna gostota	64	64
4	flesch_reading_ease_test	ocena enostavnosti branja po Fleschu	62	63
5	flesch_kincaid_grade_level	ocena nivoja Flesh-Kincaid	60	62
6	gunning_fog_index	Gunningov indeks	59	61
7	pct_of_clean_words	odstotek čistih besed	58	60
8	pct_of_punct	odstotek pojavnosti ločil	57	59
9	pct_of_stop_words	odstotek pojavnosti stop-besed	56	58
10	pct_of_direct_speech_words	odstotek besed v premem govoru	66	66
11	pct_of_quoted_words	odstotek citiranih besed	55	57
12	pct_of_sent_w_direct_speech	odstotek stavkov s premim govorom	53	55
13	pct_of_sent_w_quotes	odstotek stavkov s citiranimi besedami	52	54
14	avg_words_per_sentence	povprečno število besed na stavek	51	53
15	avg_sentences_per_paragraph	povprečno število besed na odstavek	50	52
16	avg_sentence_aoa	povprečna starost usvojitve besed v stavku	49	51
17	avg_sentence_conc	povprečna konkretnost besed v stavku	48	50
18	avg_syll_per_word	povprečno št. zlogov v besedi	47	49
19	avg_syll_per_sentence	povprečno št. zlogov v stavku	91	48
20	avg_paragraphs_per_chapter	povprečno število odstavkov na poglavje	54	56
21	avg_word_wfreq	povprečna pogostost besed	45	67
22	avg_word_lfreq	povprečna pogostost lem	67	68
23	avg_word_aoa	povprečna starost usvojitve besed	69	69
24	avg_word_conc	povprečna konkretnost besed	90	90
25	avg_sentence_lfreq	povprečna pogostost lem v stavku	89	89
26	avg_sentence_wfreq	povprečna pogostost besed v stavku	88	88
27	nr_chapters	število vseh poglavij	87	87
28	nr_paragraphs	število vseh odstavkov	86	86
29	nr_sentences	število vseh stavkov	85	85
30	nr_words	število vseh besed	84	84
31	nr_complex_words	število kompleksnih besed	83	83
32	nr_sw	število vseh stop-besed	68	82
33	nr_commas	število vejic	82	81
34	nr_words_direct_speech	število vseh besed v premem govoru	80	80
35	nr_words_quoted	število vseh citiranih besed	79	79
36	nr_words_wo_sw	število vseh besed brez stop-besed	78	78
37	nr_words_wo_punc	število vseh besed brez ločil	77	77
38	nr_words_wo_sw_and_punc	število vseh besed brez stop-besed in ločil	76	76
39	nr_punct	število vseh ločil	75	75
40	nr_of_different_lemmas	število različnih lem	72	72
41	nr_of_different_lwords	število različnih besed	71	71
42	sum_of_syllables	seštevek vseh zlogov besed	81	70
43	sentences_w_quotes	število stavkov z citiranimi besedami	44	47
44	sentences_w_direct_speech	število stavkov z premim govorom	46	91
45	nr_of_prmtvs_biti	število lem "biti"	42	46

46	nr_of_prmtvs_imeti	število lem "imeti"	20	44
47	nr_of_prmtvs_delati	število lem "delati"	19	20
48	nr_of_prmtvs_misliti	število lem "misliti"	18	19
49	S	število vseh samostalnikov	17	18
50	percent_S	delež samostalnikov	16	17
51	SL	število lastnih imen	15	16
52	percent_SL	delež lastnih imen	14	15
53	SO	število občnih imen	13	14
54	percent_SO	delež občnih imen	43	13
55	G	število vseh glagolov	12	12
56	percent_G	delež glagolov	10	11
57	P	število vseh pridevnikov	9	10
58	percent_P	delež pridevnikov	8	9
59	R	število vseh prislovov	7	8
60	percent_R	delež prislovov	6	7
61	Z	število vseh zaimkov	5	6
62	percent_Z	delež zaimkov	3	5
63	K	število vseh števnikov	2	4
64	percent_K	delež števnikov	11	3
65	D	število vseh predlogov	22	2
66	percent_D	delež predlogov	21	45
67	V	število vseh veznikov	24	22
68	percent_V	delež veznikov	41	21
69	L	število vseh členkov	40	24
70	percent_L	delež členkov	39	43
71	M	število vseh medmetov	38	42
72	percent_M	delež medmetov	37	41
73	O	število vseh okrajšav	35	40
74	percent_O	delež okrajšav	34	39
75	N	število vseh neuvrščenih besed	33	38
76	percent_N	delež neuvrščenih besed	32	37
77	sentences_with_commas_0	število stavkov brez vejic	23	36
78	sentences_with_commas_1	število stavkov z eno vejico	31	35
79	sentences_with_commas_2	število stavkov z dvema vejicama	29	34
80	sentences_with_commas_3	število stavkov s tremi vejicami	28	33
81	sentences_with_commas_4	število stavkov s štirimi vejicami	27	32
82	sentences_with_commas_5	število stavkov s petimi vejicami	30	31
83	sentences_with_commas_6	število stavkov s šestimi vejicami	92	30
84	sentences_with_commas_7	število stavkov s sedmimi vejicami	36	29
85	sentences_with_commas_8	število stavkov z osmimi vejicami	61	28
86	sentences_with_commas_9	število stavkov z devetimi vejicami	26	27
87	sentences_with_commas_10	število stavkov z desetimi vejicami	70	26
88	sentences_with_commas_11	število stavkov z enajstimi vejicami	4	25
89	sentences_with_commas_12	število stavkov z dvanajstimi vejicami	63	23
90	sentences_with_commas_13	število stavkov s trinajstimi vejicami	25	92
91	sentences_with_commas_14	število stavkov s štirinajstimi vejicami	74	74
92	sentences_with_commas_15	število stavkov z petnajstimi vejicami	73	73

Poglavje 4

Zmanjševanje dimenzionalnosti in gručenje

V tem poglavju opišemo metode zmanjševanja dimenzionalnosti, različne metrike podobnosti in gručenje. Njihovo uporabo na naših podatkih opišemo v naslednjem poglavju.

4.1 Zmanjševanje dimenzionalnosti

Metode zmanjševanja dimenzionalnosti (angl. dimensionality reduction) skušajo zmanjšati število spremenljivk tako, da dobimo čim manj nekoreliranih spremenljivk. Pri tem je treba čimbolj ohraniti strukturo podatkov, s čimer mislimo na ohranjanje razdalj med točkami. Z zmanjševanjem dimenzionalnosti se izognemo t.i. *prekletstvu dimenzionalnosti* [15, 41]. Algoritmi namreč delujejo različno dobro glede na število dimenzij in velikost vzorca [41]. Postopek ločimo na **izbiro značilk** in **izluščenje značilk**. Pri tem je pomembna **varianca** številskih spremenljivk, ki je definirana kot [15]:

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

V enačbi n predstavlja število vzorcev, μ je povprečna vrednost, ki jo izračunamo kot:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

4.1.1 Izbira značilk

S postopkom izbire značilk (angl. feature selection) ne spreminjamo vrednosti značilk, temveč nekatere izločimo iz modela. Razlog je lahko premajhna varianca (primer: vsi vzorci bakterij so iz istega planeta) ali močna korelacija dveh (primer: višina v cm in višina v inčih). V prvem primeru značilko izločimo, v drugem pa izberemo samo eno od obeh.

4.1.2 Izluščenje značilk

S postopkom izluščenja značilk (angl. feature extraction) transformiramo visokodimenzijski prostor v prostor z manj dimenzijami. Iščemo funkcijo f, ki čimbolj ohranja strukturo podatkov. Definirajmo funkcijo f_i, ki iz vseh značilk v visokodimenzijskem prostoru izračuna novo značilko E_i:

$E_i = f_i(X_1, X_2, ...X_d)$

$e = (E_1, E_2, ..., E_m)$

d – število dimenzij v visokodimenzijskem prostoru,

X – vektor značilk v visokodimenzijskem prostoru,

e – vektor značilk v nižjedimenzijskem prostoru,

m – število dimenzij v nižjedimenzijskem prostoru,

velja $m < d$.

Funkcija f je linearna ali nelinearna kombinacija značilk v višjedimenzijskemu prostoru. Postopke uporabljamo zato, da:

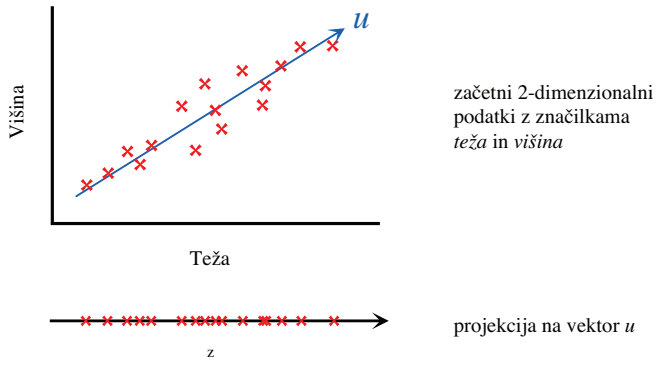
- pohitrimo hitrost nadaljnjega procesiranja,
- zmanjšamo količino prostora za shranjevanje podatkov,
- izboljšamo nekatere algoritme strojnega učenja,
- vizualiziramo podatke pri zmanjšanju na 2 ali 3 dimenzije

Primer: za optimalno uporabo algoritma k-najbližjih sosedov (angl. k-nearest neighbours) literatura predlaga zmanjšanje dimenzij na največ deset [85].

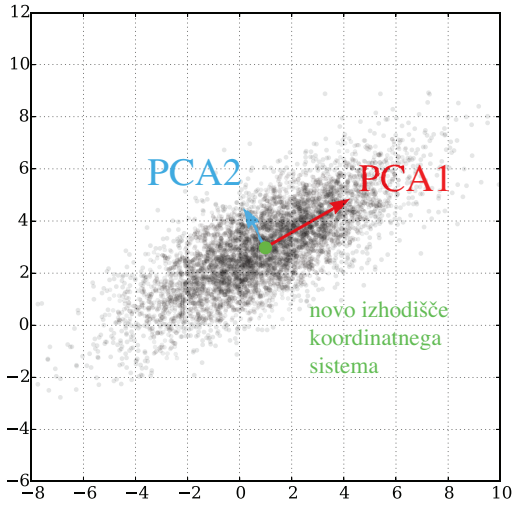
Metoda glavnih komponent

Metoda glavnih komponent (angl. principal component analysis, PCA) je linearna metoda zmanjševanja dimenzionalnosti, ki je linearna. Z njo skrčimo prostor tako, da iščemo vektor smeri, kjer je vzorec najbolj razpršen (smer največje variance), ki jo poimenujemo glavna komponenta 1 (angl. principal component 1 ali skrajšano PC1). Nato postopek ponovimo še enkrat z omejitvijo, da mora biti novi vektor ortogonalen na PC1 – ta vektor poimenujemo glavna komponenta 2 (PC2). Nato na ravnino (oz. hiperravnino), ki jo definirata vektorja, v smeri PC2 projeciramo vse točke. S tem se znebimo ene dimenzije, ohranjamo pa največjo varianco (razdaljo med točkami). Kot novo izhodišče nižjedimenzijskega prostora izberemo povprečno točko starega višjedimenzijskega. Postopek lahko ponavljamo do želenega števila dimenzij.

Postopek krčenja 2-dimenzionalnih podatkov na eno dimenzijo ilustrira slika 4.1. Glavni komponenti 1 in 2 na 2-dimenzionalnih podatkih sta prikazani na sliki 4.2.



Slika 4.1: Prikaz delovanja metode glavnih komponent pri projekciji 2-dimenzionalnih podatkov na eno dimenzijo.



Slika 4.2: Prikaz vektorjev PC1 (rdeč) in PC2 (moder) na primeru dvodimenzionalnih podatkov.

t-SNE (t-Distributed Stochastic Neighbor Embedding)

Gre za nelinearen algoritem, ki se uporablja predvsem za prikaz visokodimenzijskih podatkov v 2D in 3D prostoru [82]. Po opisu enega izmed avtorjev van der Maatena [75] je postopek boljši od ostalih algoritmov, ker se osredotoči na pravilno modeliranje majhnih razdalj med dvema točkama, lahko pa popravlja velike razlike v prostornini visokodimenzijskega prostora in njeni projekciji na 2-dimenzionalne ravnino, ki jo največkrat uporabljamo

Van der Maaten in Hinton [82] priporočata zmanjšanje dimenzionalnosti z metodo glavnih komponent na 30 dimenzij in šele nato uporabo algoritma t-SNE. S tem naj bi odpravili šum v podatkih brez pretiranih posegov v medtočkovne razdalje ter pohitrili algoritem.

4.2 Metrika

Metrika je posplošitev pojma razdalje in definira oddaljenost med elementi v dani množici. Potrebujemo jo pri algoritmih za gručenje in za izračune podobnih knjig ali uporabnikov, kjer podobnost merimo obratno sorazmerno z razdaljo v prostoru. Množici, v kateri lahko izračunavamo razdalje, pravimo tudi metrični prostor.

Metrika mora slediti določenim pravilom. Če poljubnemu paru x, y iz množice priredimo realno število $d(x, y)$, mora imeti funkcija d sledeče lastnosti:

1. $d(x, y) \geq 0$ (nenegativnost),
2. $d(x, y) = 0$, če in samo če $x = y$,
3. $d(x, y) = d(y, x)$ (simetričnost),
4. $d(x, z) \leq d(x, y) + d(y, z)$ (trikotniška neenakost).

Diskretna metrika Je najpreprostejša metrika. Definirana je z dvemi pravili:

1. $d(x, y) = 0$, če je $x = y$
2. $d(x, y) = 1$ v vseh ostalih primerih

Evklidska metrika je najpogosteje uporabljena. Če je $A = (a_1, a_2, ..., a_n)$ in $B = (b_1, b_2, ..., b_n)$, je razdalja med točkama A in B:

$$d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}$$

K-ta metrika Gre za posplošitev zgornje enačbe, kjer kvadriranje nadomestimo s k-to potenco in kvadratni koren s k-tim korenom.

$$d_k(A, B) = \sqrt[k]{|a_1 - b_1|^k + |a_2 - b_2|^k + ... + |a_n - b_n|^k}$$

Pri vrednosti k = 1 dobimo manhattansko metriko.

Metriko Tanimoto omenjamo kot zanimivost in v opomin, saj je mnogokrat zamenjana s kosinusno podobnostjo (glej 4.3).

$$T(A,B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B}$$

4.3 Kotna razdalja

Kotno razdaljo definiramo kot razliko kota dveh vektorjev, ki jih definirata točki v n-dimenzionalnem prostoru [29]. Definirana je kot:

$$d(A,B) = \frac{\arccos(\textit{kosinusna podobnost}(A,B))}{\pi}.$$

Kosinusna podobnost Kosinusna podobnost (angl. cosine similarity) je mera za podobnost med dvema vektorjema. Vektorja z enako smerjo, četudi sta različno dolga, imata vrednost kosinusne podobnosti 1. Nasprotna vektorja imata kosinusno podobnost –1, ortogonalna pa 0. Izraz izpeljemo iz skalarnega produkta:

$$A \cdot B = \|A\| \, \|B\| \cos\theta.$$

Kosinusna podobnost je torej:

$$\cos\theta = \frac{A \cdot B}{\|A\| \, \|B\|}.$$

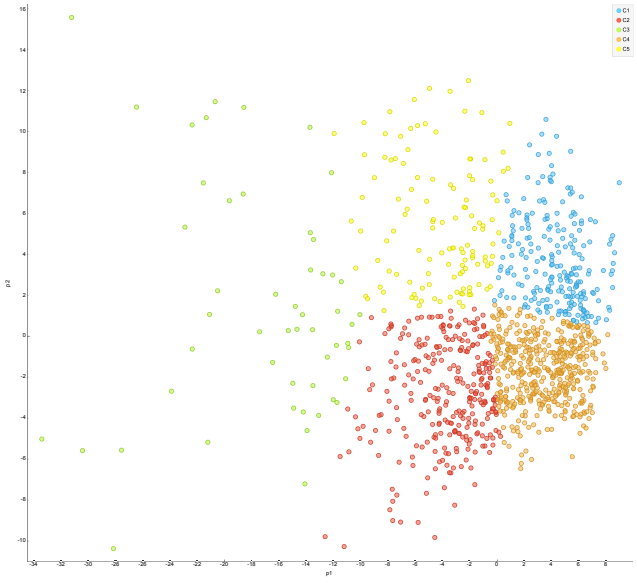
4.4 Gručenje

Gručenje (angl. clustering) je postopek nenadzorovanega strojnega učenja (glej B.2). Z njim uredimo podatke glede na njihovo sorodnost oz. podobnost. Tako dobimo gruče objektov, ki imajo skupne lastnosti, hkrati pa so drugačni od preostalih gruč. Problem gručenja je, da mora kriterije gručenja določiti uporabnik glede na namen gručenja in lastnosti podatkov.

4.4.1 Algoritem k-voditeljev

Algoritem k-voditeljev (angl. k-means) je eden najpriljubljenejših in najstarejših algoritmov za gručenje, njegov avtor je MacQueen (1967). Kot vhodni podatek potrebuje število gruč, ki ga določi uporabnik ali pa za določitev števila uporabimo oceno uspešnosti gručenja (recimo metodo Silhouette, glej 4.4.3). Metoda postavi več izhodiščnih točk (toliko, kolikor želimo imeti gruč). Izhodiščne točke so lahko izbrane naključno, boljše rezultate pa dosežemo s hevrističnim izračunom z metodo KMeans++ [34]. V prvem koraku postopek izbere točke najbližje izhodiščni točki in iz njih ustvari gručo. To naredi za vse izhodiščne točke. V drugem koraku za vsako gručo izračuna novo težiščno točko – centroid, ki je postavljen v težišče gruče in se vrne na prvi korak: iz težiščnih točk po istem postopku ustvari novo gručo. Ta dva koraka metoda ponavlja, dokler lokacija centroidov ne konvergira.

Prikaz gručenja knjig z algoritmom k-voditeljev (k = 5), s predhodnim zmanjšanjem dimenzij z metodo glavnih komponent na 2 dimenziji ilustrira slika 4.3.

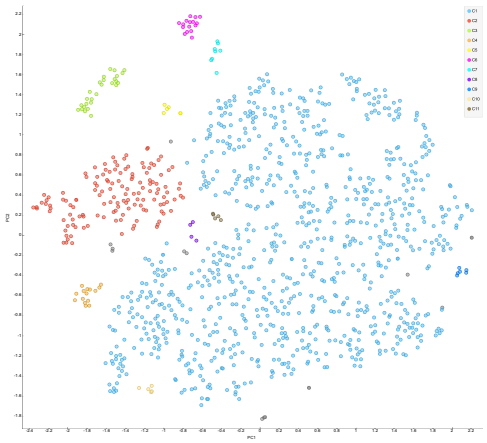


Slika 4.3: Prikaz gručenja knjig z algoritmom k-voditeljev (k=5), s predhodnim zmanjšanjem dimenzij z metodo glavnih komponent na 2 dimenziji.

4.4.2 Algoritem DBSCAN

Algoritem DBSCAN sam izbere število gruč, ni pa nujno, da gručo dodeli vsakemu objektu. Je okrajšava za skeniranje, osnovano na gostoti (angl. density based scan). Deluje torej po principu prostorske gostote, torej v gruče "zgosti" objekte iz prostora, kjer so ti dovolj gosto posejani. Točke, ležeče izven gostih področij, definira kot izjeme (angl. outlier) ter jih ne pripiše nobeni gruči. Algoritem je najpogosteje citiran algoritem za gručenje v znanstveni literaturi in je dobilnagrado *Test of Time Award* (slov. *Prestal test časa*) na vodilni konferenci za rudarjenje podatkov KDD [65] leta 2014.

Primer gručenja knjig z algoritmom DBSCAN (št. jedrnih točk = 4, razdalja = manhattanska) je prikazano na sliki 4.4. Točke predstavljajo knjige, ki jim je bila zmanjšana dimenzionalnost na 2 dimenziji s t-SNE ter ponovno z metodo glavnih komponent. Algoritem je točke razdelil v 11 gruč, 12 knjig pa je ostalo nerazporejenih in so prikazane s sivo barvo. Za tako majhen odstotek nerazporejenih knjig z metodo DBSCAN je bilo treba preizkusiti več kombinacij za zmanjševanje dimenzionalnosti in nastavitev algoritma DBSCAN. Samo gručenje je neprepričljivo, saj so točke preveč enakomerno posejane po prostoru. Zgoščenih področij, ki jih DBSCAN prepozna in poveže v gručo, praktično ni.



Slika 4.4: Prikaz gručenja knjig z algoritmom DBSCAN.

4.4.3 Metoda Silhouette

Je metoda za ocenjevanje uspešnosti gručenja [66]. Koeficient Silhouette je vrednost med 0 in 1 in določa podobnost točk v isti gruči v primerjavi z ostalimi gručami.

Za ocenjevanje moramo najprej pognati gručenje. Privzemimo, da vsaki točki pripišemo gručo (tako deluje na primer algoritem k-voditeljev). Algoritem za izračun **koeficienta Silhouete za eno točko** (p) je sledeč:

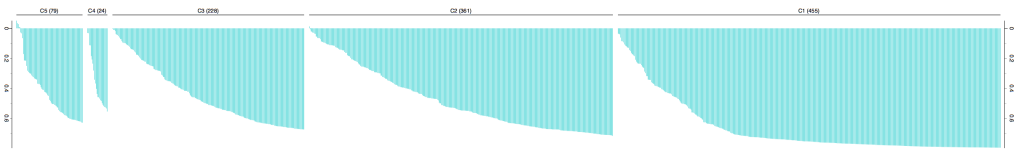
Korak 1 Za točko p v prostoru izračunamo povprečno razdaljo med vsemi preostalimi točkami znotraj iste gruče (temu recimo dA).

Korak 2 Poiščemo točki najbližjo gručo in izračunamo povprečno razdaljo med točko p in vsemi točkami najbližje gruče (temu recimo dB)

Korak 3 Koeficient Silhouette za točko p dobimo tako, da izračunamo razdaljo med dA in dB ($|dA - dB|$) in jo delimo z večjo vrednostjo izmed njih ($max(dA, dB)$).

Koeficient Silhouette za eno točko izračunamo za vse točke in rezultat povprečimo – s tem dobimo **koeficient Silhouette gručenja**. Kot najboljše izberemo tisto gručenje, ki nam da največji koeficient Silhouette gručenja.

Prikaz uspešnosti gručenja se imenuje prikaz Silhouette (angl. Silhouette plot) [64]. Primer prikaza je na sliki 4.5. Gručili smo knjige z algoritmom k-voditeljev (k = 5). Gruče so povezane z vodoravno črto z imenom gruče, medtem ko navpični stolpci prikazujejo koeficiente Silhouette za posamezne knjige.



Slika 4.5: Prikaz Silhouette na primeru gručenja knjig z metodo k-voditeljev (k=5).

Poglavje 5

Analiza podatkov

V tem poglavju predstavljamo, kako smo na naših podatkih, torej knjigah, uporabnikih in transakcijah, uporabili postopke zmanjševanja dimenzionalnosti in gručenja, ki jih opisujemo v prejšnjem poglavju.

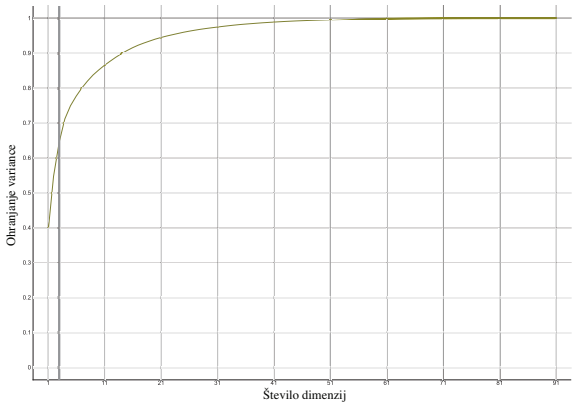
5.1 Zmanjšanje dimenzionalnosti pri knjigah

Osnovni vektor značilk knjig ima 92 dimenzij. Z metodo glavnih komponent in metodo t-SNE (za njun opis glej 4.1.2) smo zgradili nabor novih značilk in jih dodali posamezni knjigi. Nabor vektorjev značilk za vsako knjigo je zbran v tabeli 5.1.

Pri metodi glavnih komponent (skrajšano PCA) smo izbrali število k = 23 (t. i. število glavnih komponent) tako, da smo povečevali k, dokler nismo prišli do najmanjšega števila k, ki je ohranilo varianco večjo od 95 %, kot predlaga Ng [9]. Kot primer: za skok na 98 % (za 3 %) bi potrebovali dodatnih 11 dimenzij (47-% povečanje št. komponent), za skok na naslednjo, v literaturi [20] najbolj omenjano ohranitev variance, to je 99 %, bi potrebovali dodatnih 19 dimenzij (82-% povečanje št. komponent). Graf naraščanja ohranjanja variance glede na število dimenzij prikazuje slika 5.1. Rezultat sploščenja 92-dimenzionalnih podatkov knjig na dve dimenziji je prikazan na sliki 5.2.

Tabela 5.1: Vektorji značilk knjig, ki smo jih pridobili z zmanjševanjem dimenzionalnosti z metodo glavnih komponent in metodo t-SNE iz osnovnih 92 dimenzij.

ime stolpca v podatkovni bazi	metoda	novo št. dimenzij	ohranitev variance	namen zmanjševanja dimenzionalnosti
v_original	/	92	100%	osnovni vektor značilk, ki ga uporabimo pri določenih algoritmih
v_pca_23	PCA	23	95%	za klasifikacijo in gručenje
v_pca_10	PCA	10	85%	za klasifikacijo in gručenje (v literaturi priporočeno št. dimenzij za algoritem k-voditeljev)
v_pca_3	PCA	3	64%	vizualizacija v tridimenzionalnem prostoru
v_pca_2	PCA	2	54%	vizualizacija v dvodimenzionalnem prostoru
v_tsne_10	t-SNE	10	/	za klasifikacijo in gručenje (v literaturi priporočeno št. dimenzij za algoritem k-voditeljev)
v_tsne_3	t-SNE	3	/	vizualizacija v tridimenzionalnem prostoru in ponekod za klasifikacijo
v_tsne_2	t-SNE	2	/	vizualizacija v dvodimenzionalnem prostoru



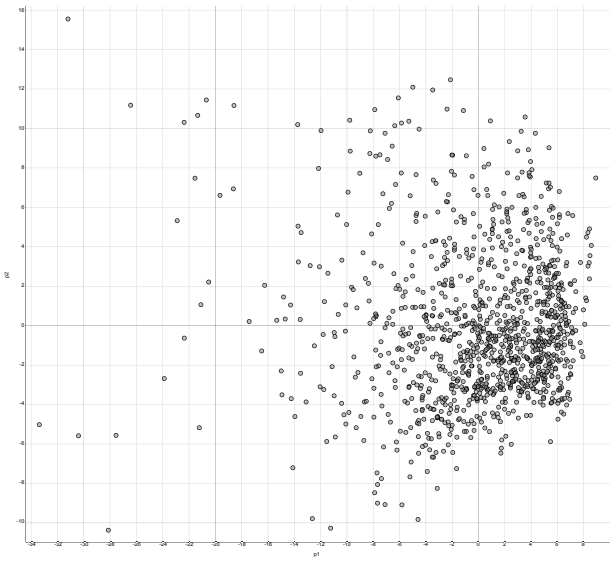
Slika 5.1: Ohranjanje variance pri metodi glavnih komponent v primeru povečevanja dimenzij na primeru knjig.

Za metodo t-SNE smo uporabili implementacijo v pythonu različice 2 [76], ki smo jo prepisali v python različice 3. Pri tem smo opazili, da je privzeta vrednost predprocesiranja z metodo glavnih komponent 50 dimenzij. Pri izračunu značilk smo upoštevali vrednost 30.

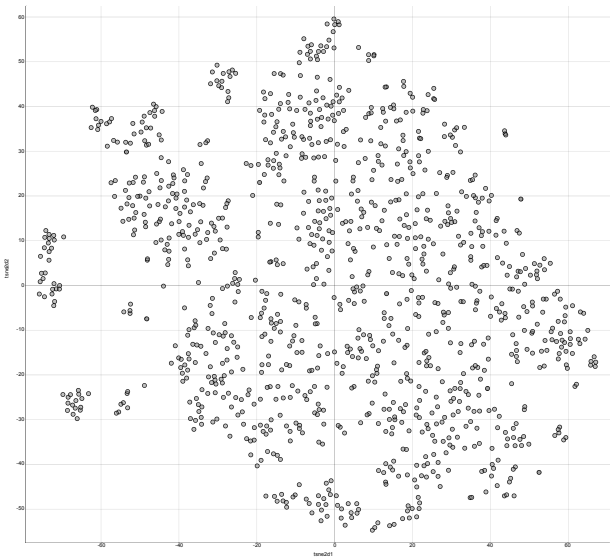
Primer uporabe t-SNE knjižnice v pythonu za 2-dimenzionalen izris knjig:

```
1 import numpy as Math
2 import pylab as Plot
3 from lib.tsne import tsne
4 X = Math.loadtxt("books.txt");
5 labels = Math.loadtxt("book_labels.txt");
6 Y = tsne(X, no_dims=2, initial_dims=30, perplexity=25.0);
7 Plot.scatter(Y[:,0], Y[:,1], 20, labels);
8 Plot.show();
```

V primerjavi z metodo glavnih komponent vidimo, da so knjige bolj razpršene po 2-dimenzionalnem prostoru, kar ilustrira slika 5.3.



Slika 5.2: Prikaz knjig v 2-dimenzionalnem prostoru z metodo glavnih komponent.

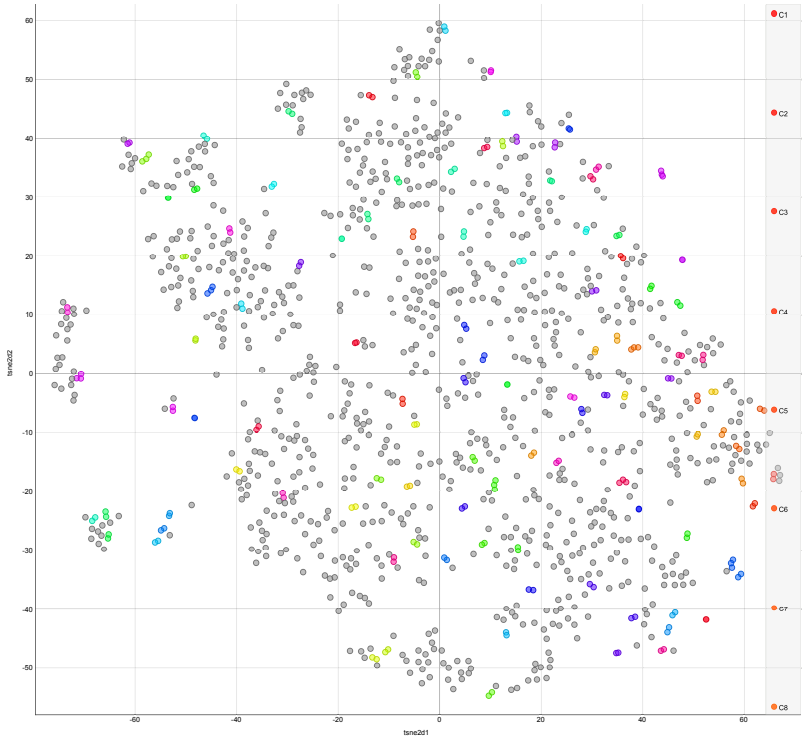


Slika 5.3: Prikaz knjig v 2-dimenzionalnem prostoru z metodo t-SNE.

5.2 Gručenje knjig

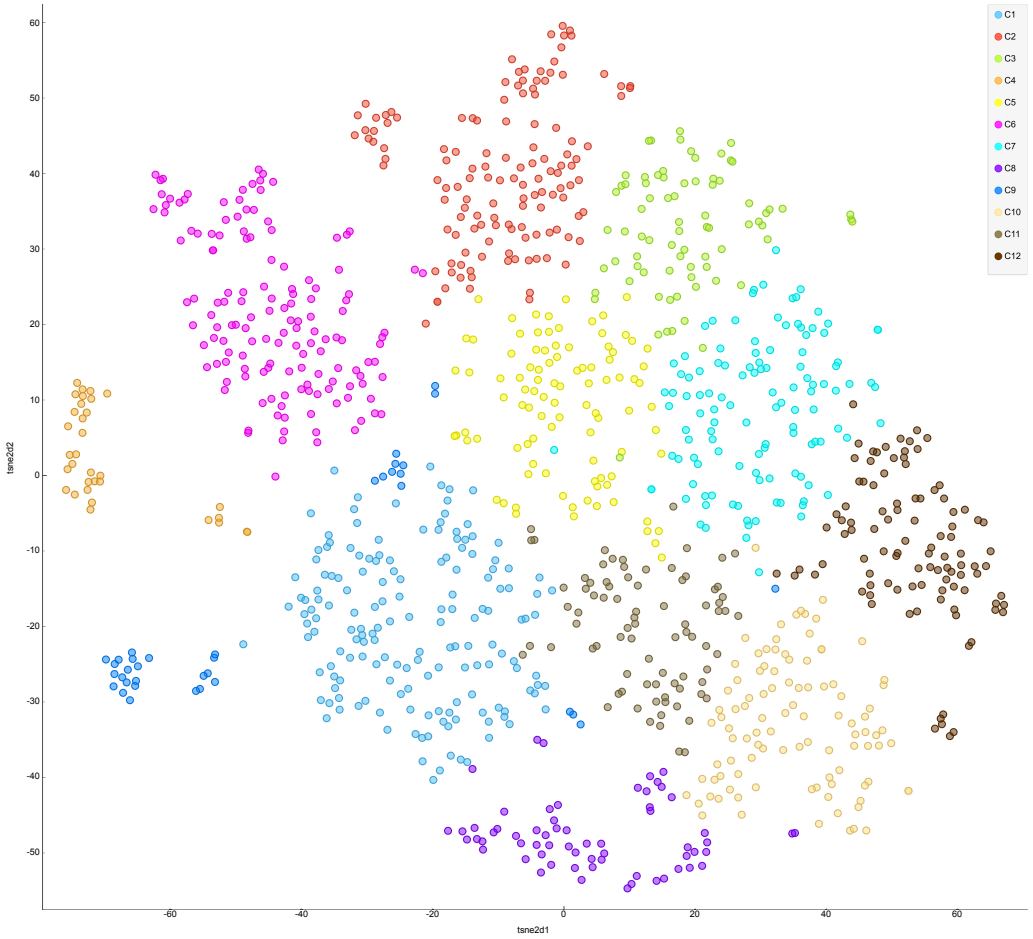
Za gručenje knjig smo preizkusili obe metodi, opisani v prejšnjem poglavju.

Gručenje z algoritmom DBSCAN Pri gručenju z algoritmom DBSCAN nismo uspeli dobiti kakovostnih in smiselnih gruč. Pri najbolj ohlapnih nastavitvah (eps = 0,9 ter število jedrnih točk = 2) smo pri najbolj sploščenih podatkih (vektor značilk fv_tsne_2) dobili 110 gruč. Osem gruč je vsebovalo tri knjige, preostalih 102 gruč pa dve. Večjih gruč ni bilo, preostalih 919 knjig ni pripadalo nobeni gruči. Zaradi predvidenega števila predlaganih knjig (glej 6) je tovrstno gručenje neuporabno in ga nismo uporabili. Gručenje prikazuje slika 5.4, pri čemer sive knjige ne pripadajo nobeni gruči.



Slika 5.4: Prikaz neuspešnega gručenja knjig z najohlapnejšimi nastavitvami algoritma DBSCAN.

Gručenje z metodo k-voditeljev Za knjige je najboljši koeficient Silhouette dosegla metoda k-voditeljev z nastavitvami k = 12 na 10-dimenzionalnem vektorju značilk, pridobljenem z metodo t-SNE. Prikaz tega gručenja knjig prikazuje slika **??**. Točke prikazujemo v 2-dimenzionalnem prostoru, definiranim z 2-dimenzionalnim vektorjem značilk, pridobljenim s t-SNE. Točke, ki ne izgledajo pravilno gručene, so si bližje v 10-dimenzionalnem prostoru, a se je del informacije izgubilo pri pretvorbi v 2-dimenzionalen prostor.



Slika 5.5: Prikaz gručenja knjig z algoritmom k-voditeljev (k=12, pridobljen z metodo Silhouette) v 10-dimenzionalnem prostoru, zmanjšanem z metodo t-SNE.

5.3 Zmanjšanje dimenzionalnosti pri uporabnikih

Kot je opisano v poglavju 3.4.3, smo pri uporabnikih izluščili dva vektorja značilk, in sicer osebni profil uporabnika z 28 značilkam ter knjižno-osebni profil uporabnika s 1.175 značilkami. Posebej slednji je zaradi visoke dimenzionalnosti težaven za nadaljnjo uporabo (glej 4.1), zato smo uporabili algoritem t-SNE. Ta opravi tudi zmanjševanje dimenzionalnosti z metodo glavnih komponent na 50 (glej 4.1.2). Zaradi tega in zaradi slabših rezultatov pri kasnejši uporabi značilk, pridobljenih z metodo glavnih komponent, smo to metodo na primeru uporabnikov opustili. Rezultat so zmanjšani vektorji značilk, predstavljeni v tabeli 5.2. Vizualizacija v dvodimenzionalnem in tridimenzionalnem prostoru je opisana v razdelku 5.4.

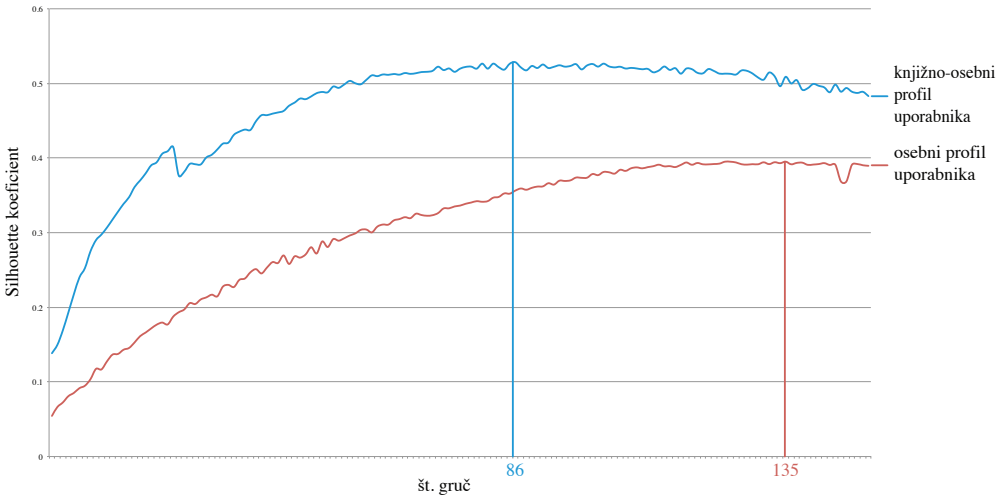
Tabela 5.2: Zmanjšani vektorji značilk uporabnikov, pridobljenih z metodo t-SNE na obeh vektorjih značilk uporabnikov.

ime stolpca v podatkovni bazi	osnovni vektor značilk	dimenzionalnost osnovnega vektorja značilk	dimenzionalnost	namen
fv_profile_tsne_10	fv_profile	28	10	izračunavanje razdalj in gručenje
fv_profile_tsne_3	fv_profile	28	3	vizualizacija v tridimenzionalnem prostoru
fv_profile_tsne_2	fv_profile	28	2	vizualizacija v dvodimenzionalnem prostoru
fv_profile_b_tsne_10	fv_profile_books	1.175	10	za izračunavanje razdalj in gručenje
fv_profile_b_tsne_3	fv_profile_books	1.175	3	vizualizacija v tridimenzionalnem prostoru
fv_profile_b_tsne_2	fv_profile_books	1.175	2	vizualizacija v dvodimenzionalnem prostoru

5.4 Gručenje uporabnikov

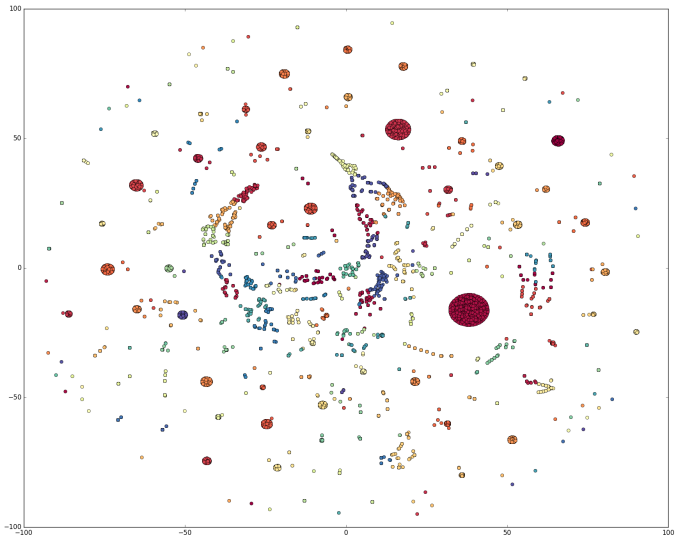
Za gručenje smo uporabili algoritem k-voditeljev. Na obeh vrstah vektorjev značilk, ki smo ju pridobili z metodo t-SNE in imata 10 dimenzij, smo pognali metodo Silhouette, s katero smo testirali kakovost gručenja s številom gruč od 2 do 150, kar ilustrira graf na sliki 5.6. Večje število gruč želimo, ker

je število uporabnikov veliko (10.813) in nam večje število gruč koristi pri optimizaciji priporočilnih algoritmov.

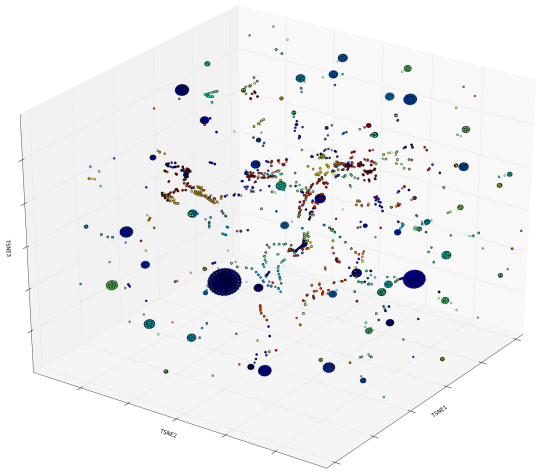


Slika 5.6: Silhouette koeficienti za gručenje uporabnikov po obeh vrstah vektorjev značilk - osebni profil uporabnika (k=135) in knjižno-osebni profil uporabnika (k=86).

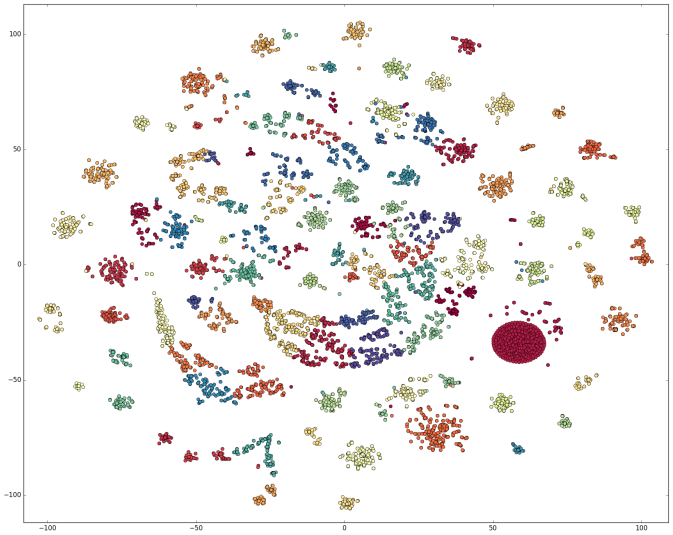
Vizualizacija uporabnikov in gruč je zanimivejša kot pri knjigah, saj so nazorno vidne gruče. Profili uporabnikov so ilustrirani na slikah 5.7 in 5.8, knjižno-osebni profili uporabnika pa na slikah 5.9 in 5.10. Na oseh grafov so vrednosti značilk, izračunane z metodo t-SNE, točke pa predstavljajo posamezne uporabnike.



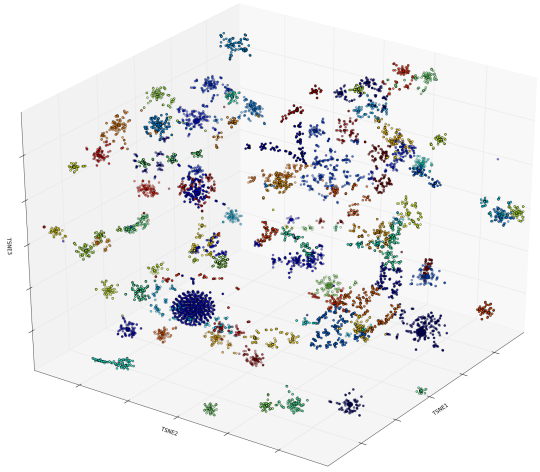
Slika 5.7: Prikaz gručenja osebnih profilov v 10 dimenzijah z algoritmom k-voditeljev(k=135), prikazan v dveh dimenzijah.



Slika 5.8: Prikaz gručenja osebnih profilov v 10 dimenzijah z algoritmom k-voditeljev(k=135), prikazan v treh dimenzijah.



Slika 5.9: Prikaz gručenja knjižno-osebnih profilov v 10 dimenzijah z algoritmom k-voditeljev(k=86), prikazan v dveh dimenzijah.



Slika 5.10: Prikaz gručenja knjižno-osebnih profilov v 10 dimenzijah z algoritmom k-voditeljev(k=86), prikazan v treh dimenzijah.

Poglavje 6

Zasnova priporočilnega sistema

V tem poglavju opišemo, kako iz podatkov, ki smo jih pripravili s postopki v prejšnjih poglavjih, zasnujemo priporočilni sistem. Na voljo imamo različne vektorje značilk, različna gručenja ter seznama pozitivnih in negativnih knjig. S kombiniranjem izbire podatkov (različnih vektorjev značilk in gručenj) ter postopkov (nastavitev vrste metrike, praga verjetnosti pozitivne klasifikacije ipd.) osnujemo več inačic priporočilnih algoritmov.

Za lažje razumevanje razložimo pojme, ki jih uporabljamo:

Seznam pozitivnih knjig uporabnika je seznam knjig, ki si jih je uporabnik izposodil, urejen po vrstnem redu izposoje. To pomeni, da je knjiga, ki si jo je uporabnik izposodil najkasneje, na dnu seznama. Iz vsake knjige v seznamu lahko pridobimo njen vektor značilk in pripadnost gruči. Skrajšano mu rečemo pozitivne knjige.

Seznam negativnih knjig uporabnika je seznam knjig, ki si jih je uporabnik ogledal samo enkrat, dodatnih transakcij s to knjigo pa ni bilo. Ta seznam zaradi nezanesljivosti (za podrobnosti glej 3.2.3) uporabimo samo pri enem algoritmu. Skrajšano mu rečemo negativne knjige.

Množica neopredeljenih knjig uporabnika vsebuje preostalo množico knjig, iz katere izbiramo kandidate v seznam predlogov. Če uporabljamo

samo seznam pozitivnih knjig uporabnika, so to vse knjige, ki niso v tem seznamu. Če uporabimo tako seznam pozitivnih kot negativnih knjig, so to vse knjige, ki niso v nobenem od teh seznamov. Tej množici skrajšano pravimo neopredeljene knjige.

Seznam predlaganih knjig je po vrstnem redu urejen seznam predlogov knjig iz množice neopredeljenih knjig uporabnika in predstavlja rezultat priporočilnega algoritma. Skrajšano mu rečemo seznam predlogov.

Število predlaganih knjig je dolžina seznama predlaganih knjig. Zaradi omejitev predvidenega uporabniškega vmesnika se odločimo predlagati $N = 10$ knjig (glej 2.3.1, kjer je število predlaganih artiklov označeno z K). V našem sistemu je to slab promil celotne množice.

6.1 Opis problema PU

Z vidika uporabnika in njegovega odnosa do knjig imamo samo en sorazmerno zanesljiv podatek, to je njegovo izposojo določene knjige. Knjige tako lahko razdelimo v dve ločeni množici:

P množica pozitivnih knjig
U množica knjig, do katerih je uporabnik neopredeljen (lahko so pozitivne ali negativne)

Vidimo, da gre za problem PU (angl. positive unclassified problem), ki spada pod enorazredno oz. unarno klasifikacijo (angl. one-class classification ali unary classification) [46]. Ta je zahtevnejša in manj zanesljiva od klasične klasifikacije.

6.2 Priporočilni algoritmi

Priporočilni algoritmi na osnovi različnih podatkov, ki jih uporabijo, vrnejo seznam priporočenih knjig. Poimenujemo jih z nosilno idejo algoritma in jih

ločimo glede na pristop (glej 2.3.2). Vsak izmed njih je označen z okrajšavo v naslovu. Osnova za primerjanje je naključni algoritem. Vsak naprednejši algoritem ima na voljo tudi rezervni algoritem, ki ga uporabimo v primeru pomanjkanja podatkov za izračun priporočil.

6.2.1 Naključni algoritem (NAK)

Naključni algoritem vrne N naključno izbranih knjig. Uporablja se kot osnova za primerjavo z ostalimi algoritmi. Če je N 10 in velikost vzorca za učenje 100, je možnost naključne izbire 10 najboljših knjig iz preostalih 1047 majhna in se giblje okoli promila. Preostali algoritmi bi po naših pričakovanjih morali dosegati precej boljše rezultate kot naključni algoritem.

6.2.2 Rezervni algoritem

Rezervni algoritem (angl. fallback algorithm) nadomesti ostale algoritme, ko ti nimajo dovolj podatkov za izračun priporočil. Če nimamo seznama pozitivnih knjig (t. i. cold start problem) niti podatkov o uporabniku, mu predlagamo najbolj priljubljene oz. najbolj izposojane knjige. Algoritem prešteje vse trenutne izposoje knjig vseh uporabnikov, jih razvrsti po padajočem vrstnem redu izposoj in prikaže prvih 10.

V primeru, da imamo le podatke o uporabniku, uporabimo kolaborativno filtriranje. Po podobnosti parametrov uporabniškega profila poiščemo najbližjega uporabnika in predlagamo knjige, ki si jih je ta izposodil. Tak pristop predlaga Lika s sod. [25]. Algoritem uporabimo v vseh spodaj naštetih algoritmih kot rezervni algoritem.

6.2.3 Algoritmi s filtriranjem vsebine

Pri algoritmih s filtriranjem vsebine za izračunavanje seznama predlogov uporabljamo izključno podatke o artiklih (torej knjigah), medtem ko podatkov o uporabnikih ne uporabimo.

Algoritem povprečne knjige (APK)

Gre za zelo enostaven algoritem, ki iz pozitvnih knjig izračuna točko, kjer bi se nahajala povprečna knjiga in izbere N knjig, najbližjih tej točki.

Algoritem je naslednji:

Korak 1 Iz seznama točk pozitivnih knjig uporabnika izračunamo težišče, torej t. i. centroid.

Korak 2 : Poiščemo najbližje knjige centroidu (po različnih metrikah) in jih prvih N po naraščajoči oddaljenosti dodamo v seznam predlogov.

Algoritem najbližjih knjig (ANB)

Algoritem se sprehodi skozi seznam pozitivnih knjig. Za vsako od knjig v seznamu izračuna najbližjo knjigo v njuni skupni gruči ter vrne seznam N najbližjih knjig. Algoritem je naslednji:

Korak 1 Izberemo seznam pozitivnih knjig.

Korak 2 Za vsako knjigo posebej izberemo iz gruče, ki ji pripada, najbližjo knjigo in jo dodamo v seznam predlogov (če knjiga že obstaja v seznamu predlogov, vzamemo naslednjo najbližjo, če knjiga gruče nima, izbiramo med vsemi knjigami).

Korak 3 Če je število predlaganih knjig premajhno (manjše od N), ponovimo koraka 1. in 2., le da v drugem koraku izberemo naslednjo najbližjo knjigo in jo dodamo v seznam predlogov.

Korak 4 Korake 1–3 ponavljamo, dokler seznam predlogov ne obsega N knjig.

Korak 5 Seznam končnih predlogov razvrstimo po naraščajoči razdalji.

Algoritem z binarno klasifikacijo (ABK)

To je edini algoritem s filtriranjem vsebine, pri katerem smo uporabili seznam negativnih knjig uporabnika. O zanesljivosti tega podatka smo že pisali (glej 3.2.3). Vseeno nas je zanimalo, ali vsebuje ta podatek napovedno moč. Dodatno težavo so predstavljali uporabniki, ki nimajo nobene negativne transakcije (takih uporabnikov je 7.299). V tem primeru je bila klasifikacija slabša. Algoritem je naslednji:

Korak 1 Na seznamu pozitivnih in negativnih knjig posameznega uporabnika poženemo učenje klasifikatorja (točneje, na vektorju značilk teh knjig).

Korak 2 Z naučenim klasifikatorjem klasificiramo množico neopredeljenih knjig uporabnika.

Korak 3 Seznam pozitivno klasificiranih knjig razvrstimo po verjetnosti pozitivne klasifikacije in jih prvih N dodamo v seznam predlogov.

Pri tem lahko uporabimo katerikoli klasifikator ali izpeljanke regresije [55], ki deluje na številskih podatkih (saj so take značilke knjig). Dodaten pogoj je, da mora klasifikator vrniti verjetnost klasifikacije. To je pomembno za razvrščanje pozitivno klasificiranih knjig, porebujemo namreč N najzanesljivejših pozitivnih klasifikacij.

Algoritem z enorazredno metodo podpornih vektorjev (APV)

Enorazredna metoda podpornih vektorjev (angl. one-class support vector machine, skrajšano one-class SVM) [43] je predlagana v literaturi [42] kot najenostavnejši primer klasifikacije PU. Gre za algoritem, namenjen ugotavljanju izstopajočih vrednosti (angl. outlier detection) [69] oziroma ugotavljanju novih vrednosti (angl. novelty detection) [70]. Gre za posebno metode podpornih vektorjev (angl. support vector machine, skrajšano SVM).

Metoda podpornih vektorjev deluje tako, da med množico binarno klasificiranih točk v prostoru postavi hiperravnino, ki točke iz ene kategorije z najširšim možnim razmakom ločuje od druge kategorije. Če točk ni mogoče ločiti z eno linearno hiperravnino, lahko za ločitev uporabimo več hiperravnin. Uporabimo lahko tudi trik jedra (angl. kernel trick), kjer množico točk, ki jih želimo klasificirati, projiciramo v drug prostor s povečanim številom dimenzij in hipperravnino izračunamo v tem prostoru.

Problema se lotimo tako, da zanemarimo množico neopredeljenih knjig. Za učenje uporabimo klasifikator OneclassSVM [69] v knjižnici scikit-learn tako, da poženemo učenje samo z vzorcem pozitivnih knjig. Klasifikacijo nato poženemo na množici neopredeljenih knjig. S tem dobimo prej neobstoječo klasifikacijo neopredeljenih knjig.

Hailong Yu s sod. [42] razlaga, da so rezultati tovrstne klasifikacije slabši kot pri klasifikaciji PU, ker pri učenju ne upoštevamo množice neopredeljenih knjig.

Algoritem ne vrača verjetnosti klasifikacije, zato smo napisali dodatno funkcijo, ki algoritem večkrat požene z različnimi pragi občutljivosti in z različnimi jedri [74] tako, da dobimo množico predlogov nabližjo številu N.

Algoritem s klasifikacijo PU (APU)

Pri klasifikaciji PU gre za to, da iz množice neopredeljenih knjig poskušamo razpoznati zanesljivo podmnožico negativnih knjig. Iz te množice nato nastane klasično nadzorovano strojno učenje.

Liu s sod. [7] povzema več algoritmov za PU klasifikacijo, ki so zbrani v tabeli 6.1.

Za ta pristop ne obstaja programska oprema ali vsaj odprtokodna knjižnica, ki bi jo lahko enostavno uporabili, našli pa smo primer kode Tanake [19], ki je povzeta po Elkan in Noto [36]. Kodo smo ustrezno predelali.

Tabela 6.1: Najpomembnejši algoritmi za PU klasifikacijo.

ime algoritma	tehnike razpoznavanja negativnih vzorcev	tehnike klasificiranja
S-EM [52]	tehnika "Spy"	EM algoritem z klasifikacijo Naivni Bayes
PEBL [53]	tehnika 1-DNF	SVM
Roc-SVM [37]	algoritem Rocchio	SVM

6.2.4 Algoritmi s kolaborativnim filtriranjem

Algoritem sorodnih uporabnikov (ASU)

Pri tem algoritmu sorodnost uporabnika izbiramo samo na podlagi podobnosti osebnega profila uporabnika. Gručenje na podlagi teh lastnosti je bilo predprocesirano in je opisano v 5.4. Algoritem je naslednji:

Korak 1 Za izbranega uporabnika poiščemo urejen seznam najsorodnejših uporabnikov, do katerih smo prišli z gručenjem na podlagi osebnega profila uporabnikov (glej 3.4.3).

Korak 2 Vzamemo seznam izposojenih knjig prvega sorodnega uporabnika ter jih dodamo na seznam predlaganih knjig v vrstnem redu izposoje sorodnega uporabnika. Če je v seznamu knjiga, ki si jo je že izposodil, jo odstranimo.

Korak 3 Če v seznamu predlaganih še ni dovolj knjig, izberemo naslednjega sorodnega uporabnika iz seznama najpodobnejših uporabnikov in ponovimo Korak 2 pri tem uporabniku.

Algoritem sorodnih transakcij (AST)

Algoritem sorodnih transakcij je enak algoritmu sorodnih uporabnikov, le da uporabimo gručenje po knjižno-osebnem profilu uporabnika – torej upoštevajoč tudi pozitivne in negativne transakcije. Algoritem je naslednji:

Korak 1 Za izbranega uporabnika poiščemo urejen seznam najpodobnejših uporabnikov, do katerih smo prišli z gručenjem na podlagi knjižno-osebnega profila uporabnikov (glej 3.4.3).

Korak 2 Vzamemo seznam izposojenih knjig prvega podobnega uporabnika ter jih dodamo na seznam predlaganih knjig v vrstnem redu izposoje sorodnega uporabnika. Če je v seznamu knjiga, ki si jo je že izposodil, jo odstranimo.

Korak 3 Če v seznamu predlaganih še ni dovolj knjig, izberemo naslednjega sorodnega uporabnika iz seznama najpodobnejših uporabnikov in ponovimo korak 2 pri tem uporabniku.

Tabela 6.2: Tabela vseh osmih priporočilnih algoritmov.

št.	**ime priporočilnega algoritma**	**okrajšava**	**vrsta**
1	Naključni algoritem	NAK	/
2	Algoritem povprečne knjige	APK	filtr. vsebine
3	Algoritem najbližjih knjig	ANB	filtr. vsebine
4	Algoritem z binarno klasifikacijo	ABK	filtr. vsebine
5	Algoritem z enoraz. met. podp. vektorjev	APV	filtr. vsebine
6	Algoritem s PU klasifikacijo	APU	filtr. vsebine
7	Algoritem sorodnih uporabnikov	ASU	kolaborativno filtr.
8	Algoritem sorodnih transakcij	AST	kolaborativno filtr.

Poglavje 7

Ovrednotenje sistema

V tem poglavju predstavimo več metod testiranja. Priporočilne algoritme ocenimo z dvema metodama. Prva je izpeljana iz obstoječih podatkov brez dodatne interakcije z uporabniki, druga pa je uporabniška študija, v kateri uporabniki ocenijo priporočilne sezname.

Obstaja več načinov za testiranje in ovrednotenje algoritmov za priporočilne sisteme. Ricci s sod. [57] nazorno razdela metodologijo, ki smo ji poskušali slediti.

Testiramo, kateri izmed več predlaganih algoritmov ima večjo natančnost predlaganja uporabnosti artikla – temu rečemo tudi "napovedna moč". Ne testiramo nobenih drugih dejavnikov (recimo hitrosti, robustnosti na izjeme, upoštevanja zasebnosti in podobno).

Pri testiranju vseh algoritmov uporabimo iste učne množice (seznam pozitivnih knjig uporabnika) in jih preverjamo na istih testnih podatkih (množica neopredeljenih knjig uporabnika).

Pri zaključkih upoštevamo, ali bo algoritem s podobno zanesljivostjo deloval tudi na novih podatkih.

7.1 Vrste testiranja

V tem podpoglavju opišemo tri različne pristope k testiranju. Prvi je **testiranje brez povezave**, kjer testiranje uporabimo že za obstoječe podatke (od tega so najpomembnejše transakcije). Drugi je **testiranje s povezavo**, kjer v delujoč sistem vpeljemo priporočanje in uporabnikom ponujamo sezname, ki so rezultat različnih priporočilnih algoritmov. Najboljšega izberemo tako, da preštejemo seznam katerega priporočilnega algoritma je bil največkrat uporabljen za želeno transakcijo (nakup, izposojo, poslušanje, ...). Ker je tovrstna vpeljava za naš prototip nemogoča, smo izbrali tretji pristop, to je **študija uporabnikov**. V študiji omejeno število povabljenih uporabnikov oceni priporočilne algoritme. Priporočilne algoritme za študijo uporabnikov nam predhodno pomaga izbrati testiranje brez povezave.

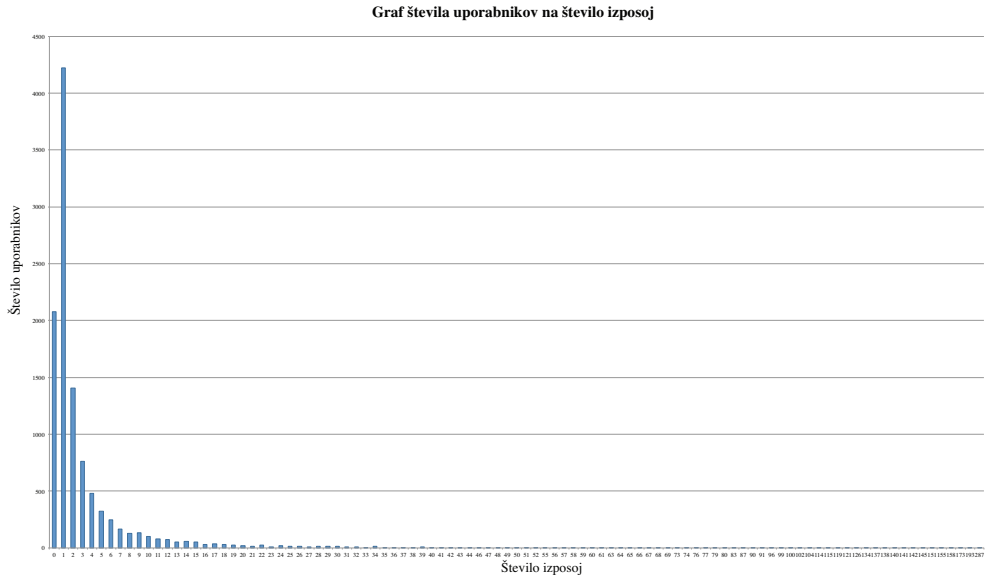
7.2 Testiranje brez povezave

Pri testiranju brez povezave (angl. offline testing) predvidevamo, da bo obnašanje uporabnika v preteklosti uporabno tudi kot model obnašanja uporabnika po uvedbi priporočilnega sistema in ne merimo neposrednega vpliva priporočilnega sistema na uporabnika. Testiranje brez povezave je po Jannach s sod. [58] v literaturi najpogostejši način testiranja priporočilnih sistemov. Ker uporabimo podatke, ki jih že imamo, ne potrebujemo dodatne interakcije z uporabniki, z njim pa lahko hitro in dokaj enotno testiramo velik nabor potencialnih algoritmov. Žal nam ta način odgovori samo na ozek nabor vprašanj, med katere sodi točnost napovedi.

7.2.1 Simulacija uporabnikovega obnašanja

Ker imamo transakcije urejene zaporedno, lahko simuliramo, kako si uporabnik skozi čas izposoja knjige. Simuliramo izposoje od prve do zadnje ter ocenjujemo, kako uspešno je sistem skozi čas predlagal knjige. Postopek deluje, če imamo na voljo mnogo podatkov, vendar je časovno potraten. Tako

temeljite analize naši podatki ne dovoljujejo, saj imamo porazdelitev števila uporabnikov glede na število izposoj porazdeljeno kot krivuljo L (znano tudi kot "dolgi rep"ali angl. long-tail) in je prikazana na sliki 7.1.



Slika 7.1: Graf števila uporabnikov glede na število izposoj.

Ricci s sod. [57] predlaga protokol določenega n (angl. given-n protocol), kjer uporabimo samo uporabnike, ki so si izposodili n knjig. Drugačna, a ekvivalentna je definicija protokola vse razen n (angl. all but n protocol), kjer fiksiramo število knjig, do katerih se nismo opredelili.

Zaradi narave podatkov smo se odločili **razdeliti uporabnike v več razredov** po številu izposoj ter na vsakem razredu uporabiti protokol določenega n. V razrede smo zbrali uporabnike s podobnim številom izposoj. Razrede smo izbrali tako, da je vzorec uporabnikov v njih čimvečji ter da pokrijemo različna števila izposoj. Pri tem nam n, torej število vseh izposoj v določenem razredu, določa tudi število izposoj za učenje. To število označimo z n_x. Z n_a označimo št. knjig v našem testnem vzorcu. Teh je $n - n_x$. Razmerje med n_x in n_a naj bo približno 80 : 20 oz. $n_x = \lfloor n * 0,8 \rfloor$ in $n_a = n - \lfloor n * 0,8 \rfloor$.

Tabela 7.1: Predstavitev razredov, v katere so razdeljeni uporabniki glede na število izposoj n.

razred	n	n_x	n_a	**št. uporabnikov v razredu**	**št. knjig, med katerimi izbira priporočilni alg.**
1	2	1	1	1.408	1.146
2	3	2	1	764	1.145
3	4	3	1	481	1.144
4	5	4	1	325	1.143
5	7	5	2	167	1.142
6	10	8	2	99	1.139
7	18	14	4	32	1.133
8	28	22	6	17	1.125
9	50	40	10	5	1.107
10	100	80	20	2	1.067

7.2.2 Testni algoritem

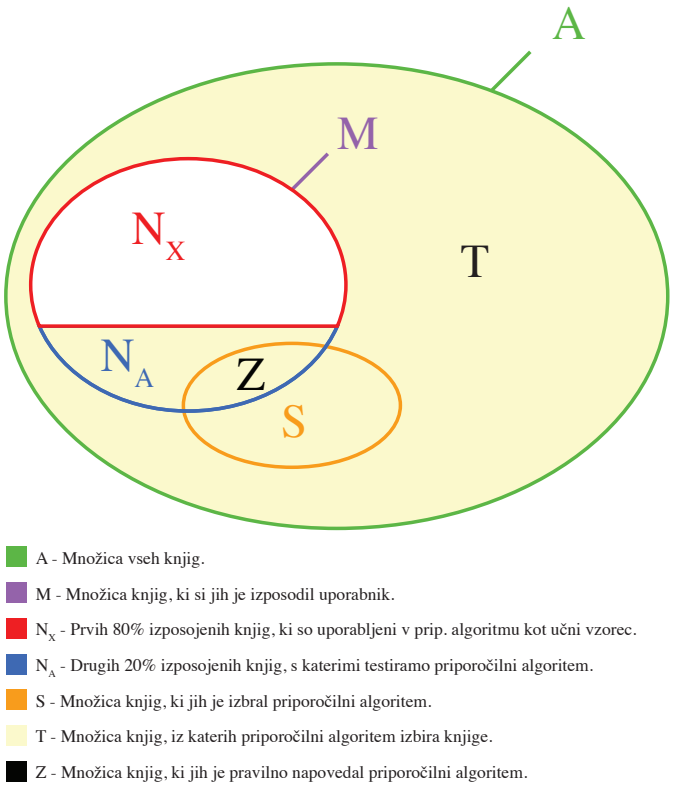
Testni algoritem je osnovan na testiranju brez povezave in je sledeč:

Korak 1 Izberemo uporabnika iz razreda X (s tem definiramo parametre n, n_x in n_a) in pridobimo seznam izposojenih knjig m.

Korak 2 Seznam m razdelimo v dve množici: prvih n_x knjig v množico N_X, vseh ostalih n_a pa v množico N_A.

Korak 3 Na množici N_X poženemo priporočilni algoritem, ki iz množice T izbere podmnožico S.

Korak 4 Iz preseka množice S in N_A (množica Z) izračunamo parametre uspešnosti priporočilnega algoritma.



Slika 7.2: Vennov diagram uporabljenih množic pri testiranju brez povezave.

Pojmi:

A množica vseh knjig,

M množica vseh knjig, ki si jih je uporabnik izposodil (dolžine n),

$\boldsymbol{N_X}$ podmnožica M, in sicer prvih n_x knjig (število n_x definira razred X),

$\boldsymbol{N_A}$ podmnožica M, in sicer preostalih n_a knjig ($n_a = n - n_x$) (množica $M - N_X$),

T množica vseh knjig, iz katerih izbiramo knjige, ki jih bomo priporočili (množica $A - N_X$),

S množica vseh knjig, ki jih izbere algoritem,

Z množica zadetkov, torej presek množice S in N_A.

Tabela 7.2: Matrika napačnih klasifikacij.

	priporočena (v mn. S)	**ni priporočena (ni v mn. S)**
izbrana od uporabnika (v mn. N_A)	pravi pozitiven (TP)	napačni negativen (FN)
neizbrana od uporabnika (ni v mn. N_A)	napačni pozitiven (FP)	pravi negativen (TN)

S tem lahko dobimo **matriko napačnih klasifikacij** (angl. missclassification matrix), ki se v literaturi pojavlja tudi pod imenom konfuzijska matrika (angl. confusion matrix) in jo ilustrira tabela 7.2.

V primeru testiranja brez povezave (ang. offline testing) knjig iz množice N_A ni predlagal priporočilni sistem, temveč jih je izbral uporabnik sam. Zato moramo pri ovrednotenju privzeti, da si uporabnik svojih neizbranih knjig ne bi izposodil, če bi bile predlagane – kakor da bi bile za uporabnika nezanimive oz. brez vrednosti. To sicer ni res, saj je lahko tudi med neizbranimi knjigami marsikatera uporabna, le da uporabnik pred priporočilnim sistemom ni imel vedenja o njej oz. mu ni bila predstavljena. Zato lahko sklepamo, da bo v količina napačnih pozitivnih napovedi ali priporočil ocenjena previsoko. Iz tabele lahko izračunamo še sledeča merila:

$$natančnost\ (angl.\ precision)\ P = TP/(TP + FP)$$

$$priklic\ (angl.\ recall)\ R = TP/(TP + FN)$$

$$stopnja\ FP = FP/(FP + TN)$$

Z dolgimi seznami priporočenih knjig se izboljša *priklic*. Ekstremen primer bi bil, da predlagamo seznam vseh preostalih knjig. Literatura [58] predlaga testiranje *natančnosti* pri različno dolgih seznamih z dolžino N, postopek imenujemo *natančnost pri* N (angl. precision at N).

V našem primeru smo se se odločili za $N = 10$ knjig (glej razdelek 6). V literaturi [57] priporočajo tudi testiranje algoritmov za različne vrednosti N.

Graf *natančnosti* in *priklica* po spreminjajočem številu N se imenuje ROC krivulja (angl. Receiver Operating Characteristic curve).

Če v izračun vključimo več uporabnikov in izračunamo povprečje *natančnosti* in *priklica* za različne N za vse uporabnike, pa to poimenujemo ROC krivulja odjemalcev (angl. customer ROC curve).

Tabela 7.3: Nastavitve testiranih različic priporočilnih algoritmov, kjer so odebeljeno označene različice algoritma z najboljšo povprečno natančnostjo.

različica algoritma / nastavitve	metrika	vektor značilk	klasifikacija	gručenje
NAK	/	/	/	/
APK1	kosin.	fv_all	/	/
APK2	evkl.	fv_all	/	/
ANB1	kosin.	fv_all	/	pca_23_km_5
ANB2	evkl.	fv_all	/	pca_23_km_5
ABK1	/	fv_pca_23	Bayesian Ridge	/
ABK2	/	fv_tsne_10	Bayesian Ridge	/
ABK3	/	fv_tsne_3	Bayesian Ridge	/
APV1	/	fv_tsne_10	OneClassSVM	/
APV2	/	fv_pca_10	OneClassSVM	/
APU1	/	fv_tsne_3	PU	/
APU2	/	fv_tsne_10	PU	/
APU3	/	fv_pca_23	PU	/
APU4	/	fv_pca_10	PU	/
ASU	evkl.	fv_profile_tsne_10	/	c_profile_tsne_10_km
AST	evkl.	fv_profile_book_10	/	c_profile_books_tsne_10_km

7.2.3 Rezultati testiranja

S spreminjanjem nastavitev (metrika, vektor značilk, klasifikacija in gručenje) smo iz osnovnih 8 priporočilnih algoritmov (glej poglavje 6.2) dobili 16 različic priporočilnih algoritmov, zbranih v tabeli 7.3, kjer so odebeljeno označene različice algoritma z najboljšo povprečno natančnostjo in jih kasneje uporabimo v študiji uporabnikov (glej 7.3). Vsako različico priporočilnega algoritma smo testirali s testnim algoritmom (glej 7.2.2) in pri $N = 10$ za vsak posamezni razred X izračunali:

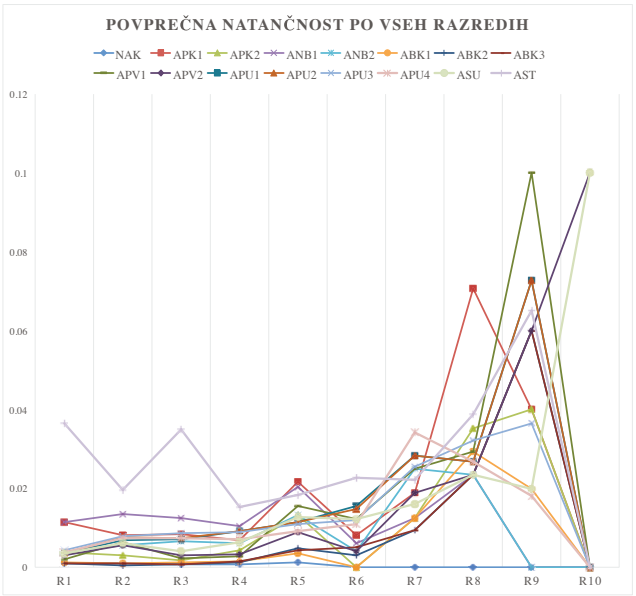
$\boldsymbol{p_a}$ je **povprečna natančnost za razred** in je seštevek izračunanih natančnosti po uporabnikih v razredu, deljen s številom uporabnikov v razredu.

$\boldsymbol{r_a}$ je **povprečni priklic za razred** in je seštevek izračunanih priklicev po vseh uporabnikih v razredu, deljen s številom uporabnikov v razredu.

Prikaz rezultatov povprečne natančnosti po razredih se nahaja v tabeli 7.4, v obliki grafa pa na sliki 7.3. Prikaz rezultatov povprečnega priklica po razredih se nahaja v tabeli 7.5, v obliki grafa pa na na sliki 7.4.

Tabela 7.4: Rezultati povprečne natančnosti (p_a) za razred po vseh algoritmih pri $N = 10$.

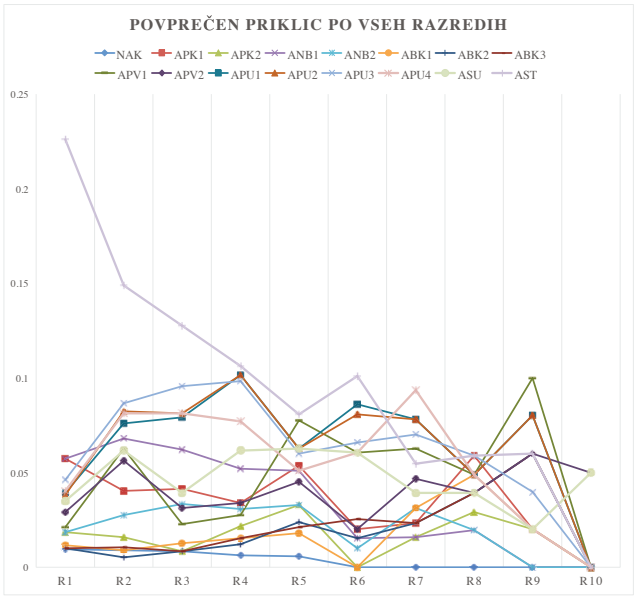
algoritem / razred	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
NAK	0,001	0,001	0,001	0,001	0,001	0,000	0,000	0,000	0,000	0,000
APK1	0,016	0,008	0,008	0,007	0,021	0,018	0,019	0,071	0,040	0,000
APK2	0,004	0,003	0,002	0,004	0,013	0,000	0,013	0,035	0,040	0,000
ANB1	0,012	0,014	0,012	0,010	0,020	0,006	0,013	0,024	0,000	0,000
ANB2	0,004	0,005	0,007	0,006	0,013	0,004	0,025	0,024	0,000	0,000
ABK1	0,001	0,002	0,001	0,002	0,004	0,000	0,013	0,029	0,020	0,000
ABK2	0,001	0,001	0,001	0,001	0,005	0,003	0,009	0,024	0,060	0,000
ABK3	0,001	0,001	0,001	0,002	0,004	0,005	0,009	0,024	0,060	0,000
APV1	0,002	0,006	0,002	0,003	0,016	0,012	0,025	0,029	0,100	0,000
APV2	0,003	0,006	0,003	0,003	0,009	0,004	0,019	0,024	0,060	0,100
APU1	0,004	0,007	0,007	0,009	0,011	0,016	0,028	0,027	0,072	0,000
APU2	0,003	0,007	0,007	0,009	0,011	0,015	0,028	0,027	0,072	0,000
APU3	0,004	0,008	0,009	0,009	0,010	0,012	0,026	0,032	0,036	0,000
APU4	0,004	0,007	0,007	0,007	0,010	0,011	0,034	0,027	0,018	0,000
ASU	0,004	0,006	0,004	0,006	0,010	0,012	0,016	0,024	0,020	0,100
AST	0,036	0,020	0,035	0,015	0,020	0,023	0,022	0,039	0,065	0,000



Slika 7.3: Graf povprečne natančnosti po razredih vseh algoritmov.

Tabela 7.5: Rezultati povprečnega priklica (r_a) za razred po vseh algoritmih pri $N = 10$.

algoritem / razred	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
NAK	0,001	0,009	0,008	0,006	0,006	0,000	0,000	0,000	0,000	0,000
APK1	0,058	0,041	0,042	0,034	0,054	0,020	0,023	0,059	0,020	0,000
APK2	0,018	0,016	0,008	0,022	0,033	0,000	0,016	0,030	0,020	0,000
ANB1	0,058	0,068	0,062	0,052	0,051	0,015	0,016	0,020	0,000	0,000
ANB2	0,018	0,027	0,033	0,031	0,033	0,010	0,031	0,020	0,000	0,000
ABK1	0,011	0,009	0,012	0,015	0,018	0,000	0,031	0,049	0,020	0,000
ABK2	0,010	0,005	0,008	0,012	0,024	0,015	0,023	0,039	0,060	0,000
ABK3	0,010	0,010	0,008	0,015	0,021	0,025	0,023	0,039	0,060	0,000
APV1	0,021	0,063	0,023	0,028	0,078	0,061	0,063	0,049	0,100	0,000
APV2	0,029	0,056	0,031	0,034	0,045	0,020	0,047	0,039	0,060	0,050
APU1	0,039	0,076	0,079	0,102	0,063	0,086	0,079	0,049	0,080	0,000
APU2	0,038	0,082	0,081	0,102	0,063	0,081	0,079	0,049	0,080	0,000
APU3	0,046	0,086	0,096	0,098	0,060	0,066	0,070	0,059	0,040	0,000
APU4	0,040	0,081	0,081	0,077	0,051	0,061	0,094	0,049	0,020	0,000
ASU	0,035	0,061	0,040	0,062	0,063	0,061	0,040	0,039	0,020	0,050
AST	0,226	0,149	0,128	0,106	0,081	0,101	0,055	0,059	0,060	0,000



Slika 7.4: Graf povprečnega priklica po razredih vseh algoritmov.

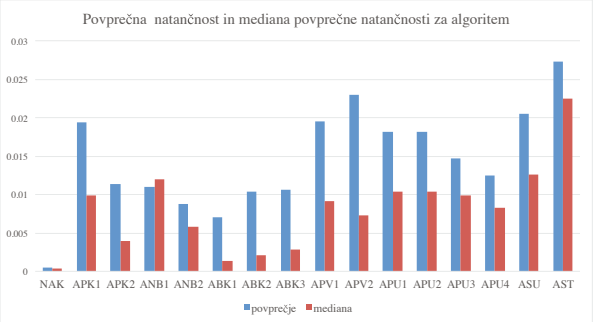
Iz obstoječih podatkov p_a in r_a smo za vsak razred lahko izračunali:

$\boldsymbol{P_a}$ je **povprečna natančnost za algoritem** in predstavlja povprečje izračunanih poprečnih natančnosti za razred za dani priporočilni algoritem.

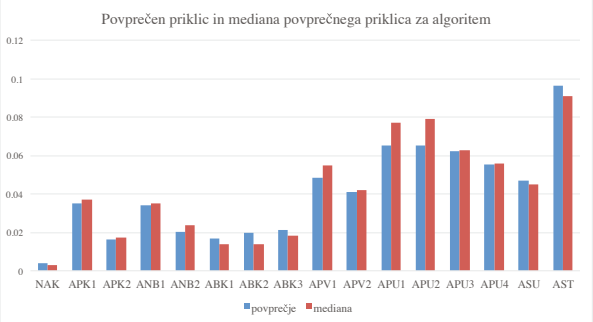
$\boldsymbol{R_a}$ je **povprečni priklic za algoritem** in predstavlja povprečje izračunanih priklicev za razred za dani priporočilni algoritem.

$\boldsymbol{MP_a}$ je **mediana povprečne natančnosti za algoritem** in predstavlja mediano izračunanih povprečnih natančnosti za razred za dani priporočilni algoritem.

$\boldsymbol{MR_a}$ je **mediana povprečnega priklica za algoritem** in predstavlja mediano izračunanih povprečnih priklicev za razred za dani priporočilni algoritem.



Slika 7.5: Graf povprečne natančnosti in mediane povprečne natančnosti.



Slika 7.6: Graf povprečnega priklica in mediane povprečnega priklica.

7.2.4 Ovrednotenje na podlagi rezultatov

Iz rezultatov je opazen nazoren razkorak med povprečno natančnostjo in povprečnim priklicem naključnega priporočilnega algoritma na skrajni levi strani grafov na slikah 7.5 ter 7.6 v primerjavi s preostalimi algoritmi. Sklepamo lahko, da nenaključni priporočilni algoritmi, ki za priporočila uporabijo več podatkov, opravljajo svojo nalogo bolje.

Možno je tudi izbrati najboljše algoritme po posameznih parametrih. Algoritem z najboljšo povprečno natančnostjo je AST, drugi je APV2 in tretji ASU. Prvi in tretji delujeta na principu kolaborativnega filtriranja, srednji pa na principu filtriranja na osnovi vsebine.

Vrstni red po priklicu je podoben, zopet je daleč najboljši AST, drugi, tretji in četrti pa so algoritmi s PU klasifikacijo. Drugi algoritem s kolaborativnim filtriranjem ASU se pri priklicu ne odreže najbolje in je uvrščen na šesto mesto.

Iz podatkov lahko sklepamo, da je algoritem AST najboljši, saj je na prvo mesto uvrščen tako po povprečni natančnosti, kot tudi po povprečnem priklicu. Zanimivo je, da algoritem ne uporablja vektorja značilk knjig, temveč podatke o izposojah drugih podobnih uporabnikov.

Pri rezultatih testiranja priporočilnih algoritmov je zanimiva majhna raznolikost rezultatov. Algoritmi so namreč realizirani zelo raznoliko in pričakovali smo večji razpon rezultatov.

Iz grafov povprečne natančnosti po razredih (slika 7.3) v segmentu malega števila podatkov o izposojenih knjigah (razredi 1–3) izstopa algoritem AST. Ker gre za edini algoritem, ki za izračun podobnih uporabnikov uporablja profile z vključenimi izposojami, je to pričakovano, saj se drugi algoritmi srečujejo z večjim pomanjkanjem podatkov. Drug algoritem s kolaborativnim filtriranjem ASU se v tem segmentu odreže samo za odtenek bolje kot ostali algoritmi s filtriranjem, osnovanim na vsebini. Očitno so podatki o izposojah, ki so pri AST vključeni v vektor značilk, pri ASU pa ne, ključni za boljše rezultate algoritma AST. V segmentu z večjim številom podatkov o izposojenih knjigah (razred 8–10) so rezultati zaradi majhnega vzorca upo-

rabnikov (vseh skupaj je 24) manj relevantni. Manjše so tudi razlike, saj se algoritmi po rezultatih povprečne natančnosti in povprečnega priklica med sabo ne razlikujejo bistveno.

7.3 Študija uporabnikov

Študija uporabnikov za razliko od testiranja brez povezave (glej 7.2) zahteva interakcijo z uporabniki. Izbrali smo osem različnih algoritmov z nastavitvami, ki so pri testiranju brez povezave dosegli najobetavnejše rezultate in so v tabeli 7.3 označeni odebeljeno. V študiji uporabnikov uporabljamo njihove okrajšave brez različice. Testirali smo na vzorcu devetih testnih uporabnikov, katerih lastnosti prikazuje tabela 7.6. Povprečna letnica rojstva je 1974, 5 jih je moškega in 4 ženskega spola. Povprečna izobrazba je dokončana višja šola. Večina je zaposlenih in podjetnikov. Od njih smo pridobili sledeče podatke:

- spol,
- letnica rojstva,
- kategorija,
- izobrazba,
- seznam 10 knjig, ki bi si jih izposodili oz. so želene,
- seznam 5 knjig, ki jih ne bi izposodili oz. so neželene.

Pridobljene podatke smo uporabili za izračun seznamov priporočenih knjig, ki jih dajejo različni priporočilni algoritmi. Dobljene sezname so uporabniki pregledali in določili vrstni red seznamov po njihovi oceni. Vrstni red seznamov smo uporabili za razvrstitev algoritmov od najboljšega do najslabšega.

Uporabniška izbira želenih in neželenih knjig Ker so priporočilni algoritmi prototipni in delujejo samo na razvojnem računalniku, smo izdelali ločen uporabniški vmesnik za izbiro knjig. Uporabnik je iz spustnega menija izbral 10 knjig, ki so mu všeč oz. bi si jih izposodil, ter 5 knjig, ki mu

Tabela 7.6: Lastnosti testnih uporabnikov, ki so bile potrebne za algoritme s kolaborativnim filtriranjem.

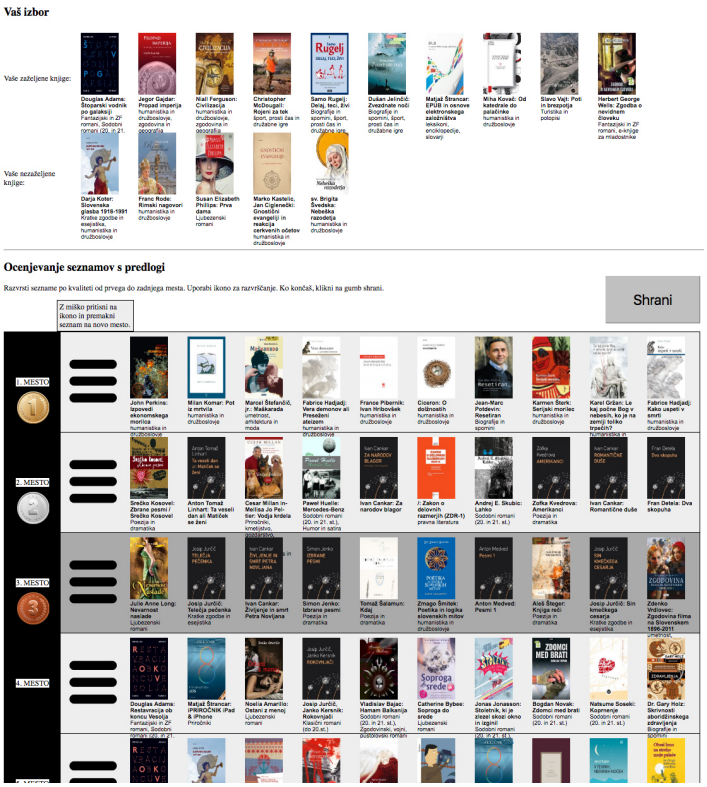
letnica rojstva	spol	izobrazba	kategorija
1982	M	7	8
1976	Z	6	8
1984	Z	8	6
1982	M	5	8
1964	Z	5	10
1955	M	6	6
1993	M	5	5
1969	Z	5	6
1963	M	5	6

niso všeč oz. si jih ne bi. Podrobnosti knjig si je lahko ogledal v obstoječi elektronski knjižnici. Pri tem je bil opozorjen, da če knjige ni v spustnem meniju, si je ne more izbrati, tudi če obstaja v elektronski knjižnici. S tem smo izločili možnost, da si izbere knjigo v formatu PDF, ki jih nimamo v naši zbirki podatkov (glej 3.2.1).

Izdelava priporočilnih seznamov Izbrane knjige smo uporabili kot vhodne podatke za priporočilne algoritme. Uporabnike smo dodali v podatkovno bazo, jim ustvarili oba osnovna vektorja značilk in znova pognali algoritem t-SNE z želeno dimenzionalnostjo 10. Gručenja uporabnikov nismo ponovili. Algoritma s kolaborativnim filtriranjem, ki te podatke potrebujeta, delujeta namreč tudi brez gručenja. Ker morata izračunati matriko razdalj med vsemi uporabniki in ne samo med uporabniki v gruči, traja izračun dlje časa. Rezultate priporočilnih algoritmov smo prenesli nazaj v ločen uporabniški vmesnik v obliki priporočilnih seznamov.

Uporabniško ocenjevanje priporočilnih seznamov Uporabniku smo poslali unikatno spletno povezavo, na kateri so bili prikazani seznami prej iz-

branih všečnih ali nevšečnih knjig. Spodaj je bilo navpično razvrščenih vseh 8 priporočilnih seznamov v naključno izbranem vrstnem redu. Vsak seznam je vseboval 10 vodoravno razvrščenih knjig, ki jih je vrnil priporočilni algoritem. Sezname so lahko uporabniki s principom "povleci in spusti"poljubno razvrstili od prvega do zadnjega mesta. S klikom na naslovnico knjige so si lahko ogledali podrobnosti knjige. Po končanem razvrščanju so kliknili gumb Shrani. Možno je bilo tudi večkratno shranjevanje. Za boljšo nazornost so bila prva tri mesta označena z medaljami. Testnim uporabnikom smo razložili, da naj poskusijo oceniti seznam kot celoto in ne samo prvih nekaj knjig s seznama. Zaslon spletnega uporabniškega vmesnika za ocenjevanje priporočilnih seznamov je prikazan na sliki 7.7. V prikazanem primeru uporabnik še ni razvrstil seznamov.



Slika 7.7: Zajem zaslona spletnega uporabniškega vmesnika za uporabniško ocenjevanje priporočilnih seznamov.

Problematizacija Ker testni uporabniki razvrščajo po vrstnem redu, so lahko določene odločitve, kateri seznam je boljši, tesnejše kot druge. Recimo prvi algoritem je lahko zanje precej boljši od drugega ali tretjega, medtem ko je drugi boljši od tretjega le za odtenek. Za točnejše meritve bi potrebovali še dodatno oceno, za koliko je kakšen seznam boljši od drugega. Ker bi to dodatno otežilo uporabniški vmesnik in razumevanje ocenjevanja, se tega nismo lotili. Iz razvrščanja ne izvemo razloga, zakaj je nek seznam uvrščen boljše kot drugi. Uporabniku namreč ni potrebno argumentirati, zakaj je en seznam uvrščen višje kot drug. Odločitev za določen vrstni red je lahko na primer ena neustrezna knjiga za slabše uvrščen seznam, medtem ko je odločitev za boljšo uvrstitev nekega seznama lahko le ustrezno predlagana prva knjiga. Od uporabnikov smo dobili povratno informacijo, da je predvsem razvrščanje nižjih mest težavnejše, saj so seznami manj ustrezni, medtem ko razvrščanje na prva tri mesta ni težavno. Za natančnejšo analizo bi bilo treba povečati vzorec in vanj vključiti širši spekter različnih profilov, predvsem starejših in manj izobraženih uporabnikov, po možnosti iz slabo zastopanih kategorij (šolarji, upokojenci ipd.).

Rezultati so prikazani v tabeli 7.7. Rezultate razvrstitev smo statistično obdelali in so prikazani v tabeli 7.8.

Evaluacija rezultatov Iz rezultatov je moč sklepati, da so uporabnikom naljubši algoritmi s kolaborativnim filtriranjem, saj sta oba tovrstna algoritma na prvih mestih. Upoštevajoč nižjo standardno deviacijo (stolpec σ) uvrstitve sta konsistentno ocenjena. Na tretjem mestu je najenostavnejši algoritem za filtriranje na osnovi vsebine, kar dokazuje, da je moč narediti delujoč priporočilni sistem na podlagi stilometričnih značilk knjig. Četrto mesto zaseda algoritem s klasifikacijo PU, na podlagi česar bi lahko sklepali, da obstoječa rešitev klasifikacije PU deluje. Peto mesto zaseda algoritem povprečne knjige, česar ne znamo pojasniti in gre lahko za naključje. Ta algoritem daje pri raznovrstnejših knjigah slabe rezultate. Za razjasnitev bi bilo treba analizirati raznolikost zaželenih knjig testnih uporabnikov, ki so ta

algoritem uvrstili višje. Algoritem z binarno klasifikacijo ima očitno premalo podatkov za učenje, vendar je še vedno boljši od zadnjega uvrščenega algoritma, ki ne upošteva negativnih knjig. Izstopa ocena naključnega algoritma, ki se ni uvrstil na zadnje, ampak predzadnje mesto, iz česar sklepamo, da bi v produkcijski različici lahko določen odstotek knjig predlagali naključno. Najslabše je bil ocenjen algoritem z enorazredno metodo podpornih vektorjev, nezanesljivost metode omenjajo že predlagatelji (glej 6.2.3).

Tabela 7.7: Razvrstitev seznamov priporočilnih algoritmov po uporabnikih.

up. / alg.	**NAK**	**APK**	**ANB**	**ABK**	**APV**	**APU**	**ASU**	**AST**
1	5	7	3	6	8	4	1	2
2	8	4	2	6	7	5	3	1
3	6	5	2	8	7	4	2	3
4	8	7	3	4	6	5	1	2
5	5	7	2	4	6	8	1	3
6	8	2	6	7	5	3	4	1
7	6	4	1	8	7	5	3	2
8	5	3	4	8	7	6	2	1
9	6	7	4	1	8	5	2	1

Tabela 7.8: Rezultati razvrstitve priporočilnih algoritmov s strani testnih uporabnikov.

uvrstitev	**algoritem**	**povprečna uvrst.**	**mediana uvrst.**	**σ uvrst.**
1	AST	1,8	2	1,32
2	ASU	2,1	2	1,97
3	ANB	3,0	3	1,5
4	APU	5,0	5	3,39
5	APK	5,1	5	0,97
6	ABK	5,8	6	1,41
7	NAK	6,3	6	1,05
8	APV	6,8	7	0,83

Poglavje 8

Sklepne ugotovitve

V diplomski nalogi smo obravnavali področje priporočilnih sistemov in izdelali prototip za potrebe izposoje v e-knjižnici. Izračunali smo številske stilometrične značilnosti besedil, izluščenih iz množice e-knjig v formatu ePub. Pri tem smo uporabili strojno učenje na korpusu ccGigafida ter izbrali najhitrejši in najboljši klasifikator za potrebe oblikoslovnega označevanja besedil in lematizacije. Za nadaljnjo pripravo podatkov smo uporabili zmanjševanje dimenzionalnosti, različna gručenja in tudi različne metrike za izračunavanje razdalj med knjigami in uporabniki. Pri tem smo upoštevali specifičnost podatkov in razrešili probleme, ki jih prinašajo manjkajoče vrednosti in imenski atributi. Na podlagi podatkov o uporabnikih, njihovih izposojah in obiskih strani smo osnovali več priporočilnih algoritmov za priporočanje knjig, osnovanih na kolaborativnem filtriranju in filtriranju na osnovi vsebine. Algoritme smo testirali in jih ovrednotili.

Pokazali smo, da je možno izdelati priporočilni sistem za izposojo e-knjig samo z uporabo stilometričnih lastnosti knjig. Sklepamo, da bi bila v primeru vključitve dodatnih lastnosti knjig kakovost priporočil večja. Testiranje in ovrednotenje priporočilnih algoritmov je pokazalo, da algoritmi s kolaborativnim filtriranjem dosegajo boljše rezultate kot algoritmi s filtriranjem, osnovanim na vsebini.

V nadaljnjem delu bi bilo smiselno izdelati hibriden algoritem, torej pri-

poročilni sistem, ki bi združil pozitivne lastnosti obeh pristopov. Za produkcijsko različico bi bilo za večjo natančnost smiselno vključiti tudi ostale podatke o knjigi, recimo avtorja, založbo, leto izdaje, različne klasifikacije (recimo CIP) itd. Pomembno izboljšavo bi lahko prinesla analiza vsebine, ki bi vektorje utežila z metodo *tf-idf* [77]. Lahko bi dodali tudi več stilometričnih značilk, recimo bigrame in trigrame redkih besednih zvez, ki jim v literaturi pripisujejo precejšnjo pomembnost za določanje stila pisanja.

V sistem bi lahko dodali tudi binarno ocenjevanje knjig, kot na primer "všeč mi je"in "ni mi všeč"ter s tem pridobili zanesljivejše eksplicitne podatke. S tem bi poenostavili algoritme, saj bi problem iz težavne PU klasifikacije prevedli na binarno klasifikacijo. Ocenjujemo, da bi s tem lahko močno izboljšali kakovost priporočil.

Izdelan prototip bi lahko z manjšimi popravki uvedli v slovenske e-knjižnice kot seznam priporočenih knjig za posameznega uporabnika ali za sezname podobnih knjig na straneh s podrobnostmi o knjigi, kot je to navada pri večjih spletnih knjigarnah. Lahko bi pridobili podatke o izposojah fizičnih knjig v knjižnicah ter uporabili kolaborativno filtriranje, ki analize knjig ne potrebuje. Izračunani priporočilni seznami bi bili lahko dober pripomoček knjižničarjem za priporočanje knjig uporabnikom, ki e-knjižnic še ne uporabljajo.

Literatura

[1] RecSys Community. "The ACM Conference Series on Recommender Systems" [Online]. Dosegljivo: `https://recsys.acm.org/`. [Dostopano 5. 5. 2016].

[2] V. Kuperman, H. Stadthagen-Gonzalez, M. Brysbaert, "Age-of-acquisition ratings for 30,000 English words", *Behavior Research Methods*, št. 4, zv. 44, str. 978—990, 2012.

[3] *AI Communications - Recommender Systems*, apr. 2008, št. 2—3, zv. 21.

[4] scikit-learn developers. "Bayesian Ridge Regression" [Online]. Dosegljivo: `http://scikit-learn.org/stable/modules/linear_model.html#bayesian-ridge-regression` [Dostopano 16. 5. 2016].

[5] M. Kuharič, "Berljivost pisma predsednika uprave v letnih poročilih slovenskih javnih družb", Diplomsko delo, EPF - Ekonomsko-poslovna fakulteta, Univerza v Mariboru, Maribor, 2015.

[6] MongoDB. "BSON - Specification Version 1.0" [Online]. Dosegljivo: `http://bsonspec.org/spec.html` [Dostopano 2. 5. 2016].

[7] B. Liu, Y. Dai,X. Li, W.S. Lee, P.S. Yu, "Building text classifiers using positive and unlabeled examples", v zborniku *Data Mining, 2003. ICDM 2003. Third IEEE International Conference*, Melbourne, Florida, ZDA, nov. 2003, str. 179—186.

[8] B. Loenneker, P. Jakopin, "Checking POSBeseda, a Part-of-Speech tagged Slovenian corpus", v zborniku *Zbornik 7. mednarodne multikonference Informacijska družba IS 2004*, Ljubljana, Slovenija, okt. 2004, str. 48—55.

[9] A. Ng. "Choosing the number of Principal Components" [Online]. Dosegljivo: `https://www.coursera.org/learn/machine-learning/lecture/S1bq1/choosing-the-number-of-principal-components` [Dostopano 8. 5. 2016].

[10] L. Feng, N. Elhadad, M. Huenerfauth, "Cognitively motivated features for readability assessment", v zborniku *EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Atene, Grčija, 2009, str. 229—237.

[11] Wikipedia. "Comma-separated values" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Comma-separated_values` [Dostopano 2. 5. 2016].

[12] M. Brysbaert, A.B. Warriner, V. Kuperman, "Concreteness ratings for 40 thousand generally known English word lemmas", *Behavior Research Methods*, št. 3, zv. 46, str. 904—911, 2014.

[13] Wikipedia. "Contingency table" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Contingency_table` [Dostopano 26. 6. 2016].

[14] Wikipedia. "Cosine similarity" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Cosine_similarity` [Dostopano 28. 5. 2016].

[15] Wikipedia. "Curse of Dimensionality" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Curse_of_dimensionality` [Dostopano 26. 5. 2016].

[16] Wikipedia. "Data extraction" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Data_extraction` [Dostopano 11. 5. 2016].

[17] Wikipedia. "Datotečni format ZIP" [Online]. Dosegljivo: `https://sl.wikipedia.org/wiki/Datotečni_format_ZIP` [Dostopano 27. 4. 2016].

[18] IJS. "Definicije oblikoskladenjskih kategorij" [Online]. Dosegljivo: `http://nl.ijs.si/jos/msd/html-sl/msd.specs.html` [Dostopano 11. 5. 2016].

[19] H. Tanaka. "A demo code for PU classification proposed by Elkan and Noto 2008" [Online]. Dosegljivo: `https://gist.github.com/nkt1546789/e9421f06ea3a62bfbb8c` [Dostopano 28. 6. 2016].

[20] M. Blum. "Dimensionality Reduction" [Online]. Dosegljivo: `http://ml.informatik.uni-freiburg.de/_media/documents/teaching/ss12/ml/13_pca.pdf` [Dostopano 28. 5. 2016].

[21] Zbornik *E-Commerce and Web Technologies, 13th International Conference, EC-Web 2012*, Dunaj, Avstrija, sept. 2012.

[22] Aleksandar Erkalović. "ebooklib" [Online]. Dosegljivo: `https://github.com/aerkalov/ebooklib` [Dostopano 25. 4. 2016].

[23] Wikipedia. "EPUB" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/EPUB`. [Dostopano 9. 5. 2016].

[24] M.Balabanović, Y. Shoham, "Fab: content-based, collaborative recommendation", *Communications of the ACM*, mar. 1997, št. 3, zv. 40, str. 66—72.

[25] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, "Facing the cold start problem in recommender systems", *Expert Systems with Applications*, mar. 2014, št. 4, zv. 41, str. 2065—2073.

[26] Wikipedia. "Flesch reading ease" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch_reading_ease` [Dostopano 15. 4. 2016].

[27] Wikipedia. "Flesch–Kincaid grade level" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch.E2.80.93Kincaid_grade_level [Dostopano 15. 4. 2016].

[28] Wikipedia. "Gunning fog index" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Gunning_fog_index [Dostopano 15. 4. 2016].

[29] *IEEE Intelligent Systems*, maj—jun. 2007, št. 3, zv. 22.

[30] IDPF. "International Digital Publishing Forum" [Online]. Dosegljivo: http://idpf.org/ [Dostopano 3. 5. 2016].

[31] *International Journal of Electronic Commerce*, zima 2006—2007, št. 2, zv. 11.

[32] Coursera Inc. "Introduction to Recommender Systems" [Online]. Dosegljivo: https://www.coursera.org/learn/recommender-systems [Dostopano 31. 5. 2016].

[33] Wikipedia. "JSON" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/JSON [Dostopano 5. 5. 2016].

[34] Wikipedia. "K-means++" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/K-means++ [Dostopano 18. 5. 2016].

[35] X. He, D. Cai, P. Niyogi, "Laplacian score for feature selection", *Advances in neural information processing systems 18 (NIPS 2005)*, 2005, str. 507—514.

[36] C. Elkan, K. Noto, "Learning classifiers from only positive and unlabeled data", v zborniku *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining 2008*, Las Vegas, ZDA, avg. 2008, str. 213—220.

[37] X. Li, B. Liu, "Learning to classify texts using positive and unlabeled data", v zborniku *IJCAI 2003*, Acapulco, Mehika, avg. 2003, zv. 3, str. 587—592.

[38] M. Juršic, I. Mozetic, T. Erjavec, N. Lavrac, "Lemmagen: Multilingual lemmatisation with induced ripple-down rules", *Journal of Universal Computer Science*, maj 2010, št. 9, zv. 16, str. 1190—214.

[39] Wikipedia. "Lexical density" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Lexical_density` [Dostopano 14. 6. 2016].

[40] H. Daume III, "Mega model optimization package" [Online]. Dosegljivo: `http://www.umiacs.umd.edu/~hal/megam/` [Dostopano 14. 5. 2016].

[41] scikit-learn developers. "Nearest neighbor and the curse of dimensionality" [Online]. Dosegljivo: `http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html#nearest-neighbor-and-the-curse-of-dimensionality` [Dostopano 15. 5. 2016].

[42] H. Yu, W. Zuo, T. Peng, "A new PU learning algorithm for text classification", v zborniku *Mexican International Conference on Artificial Intelligence*, Monterrey, Mehika, nov. 2005, str. 824—832.

[43] scikit-learn developers. "Novelty and Outlier Detection" [Online]. Dosegljivo: `http://scikit-learn.org/stable/modules/outlier_detection.html` [Dostopano 24. 5. 2016].

[44] Wikipedia. "Novelty detection" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Novelty_detection` [Dostopano 24. 5. 2016].

[45] M. Grčar, S. Krek, K. Dobrovoljc, "Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik", v zborniku *Zbornik Osme konference Jezikovne tehnologije*, Ljubljana, Slovenia, okt. 2012.

[46] Wikipedia. "One-class classification" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/One-class_classification [Dostopano 14. 5. 2016].

[47] Wikipedia. "One-hot" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/One-hot [Dostopano 28. 5. 2016].

[48] scikit-learn developers. "OneHotEncoder" [Online]. Dosegljivo: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html [Dostopano 7. 5. 2016].

[49] Biolab. "Orange Imputation Model" [Online]. Dosegljivo: http://docs.orange.biolab.si/reference/rst/Orange.feature.imputation.html#model-based-imputation [Dostopano 11. 5. 2016].

[50] Wikipedia. "Outlier detection" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Outlier#Detection [Dostopano 24. 5. 2016].

[51] B. Schwartz, *The Paradox of Choice*, New York: Harper Perennial, 2004.

[52] B. Liu, W.S. Lee, P.S. Yu, X. Li, "Partially supervised classification of text documents", v zborniku *International Conference on Machine Learning 2002*, Melbourne, Avstralija, jul. 2002, zv. 2, str. 387—394.

[53] H. Yu, J. Han, K.C. Chang, "PEBL: positive example based learning for web page classification using SVM", v zborniku *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Kanada, jul. 2002, str. 239—248.

[54] Microsoft. "Preskušanje berljivosti dokumenta" [Online]. Dosegljivo: https://support.office.com/sl-si/article/Preskusanje-berljivosti-dokumenta-85B4969E-E80A-4777-8DD3-F7FC3C8B3FD2 [Dostopano 2. 6. 2016].

[55] Wikipedia. "Recommender Systems" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Recommender_system. [Dostopano 5. 5. 2016].

[56] (2012) L. Xiang, "Recommender System Introduction" [Online]. Dosegljivo: `http://www.slideshare.net/xlvector/recommender-system-introduction-12551956` [Dostopano 7. 5. 2016].

[57] F. Ricci, P.B. Kantor, L. Rokach, B. Shapira, *Recommender systems handbook*, New York: Springer US, 2011.

[58] D. Jannach, M. Zanker, M. Ge, M. Gröning, "Recommender systems in computer science and information systems–a landscape of research", v zborniku *International Conference on Electronic Commerce and Web Technologies 2012*, sep. 2012, zv. 123, str. 76—87.

[59] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, *Recommender systems: an introduction*, Cambridge: Cambridge University Press, 2010.

[60] T. Erjavec, N. Logar Berginc, "Referenčni korpusi slovenskega jezika (cc)Gigafida in (cc)KRES", v zborniku *Zbornik Osme konference Jezikovne tehnologije*, Ljubljana, Slovenija, okt. 2012, str. 57—62.

[61] A. Narayanan, V. Shmatikov, "Robust de-anonymization of large sparse datasets", v zborniku *2008 IEEE Symposium on Security and Privacy (sp 2008)*, Oakland, Kalifornija, ZDA, maj 2008, str. 111—125.

[62] scikit-learn developers. "scikit-learn: Machine Learning in Python" [Online]. Dosegljivo: `http://scikit-learn.org/` [Dostopano 27. 4. 2016].

[63] J. W. Pennebaker, "The Secret Life of Pronouns: What Our Words Say About Us", Bloomsbury USA, 2011.

[64] scikit-learn developers. "Selecting the number of clusters with silhouette analysis on KMeans clustering" [Online] Dosegljivo: `http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html` [Dostopano 3. 6. 2016].

[65] SIGKDD. "2014 SIGKDD Test of Time Award" [Online]. Dosegljivo: http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award [Dostopano 5. 5. 2016].

[66] Wikipedia. "Silhouette clustering" [Online] Dosegljivo: https://en.wikipedia.org/wiki/Silhouette_(clustering) [Dostopano 3. 6. 2016].

[67] scikit-learn developers. "sklearn.feature_extraction.DictVectorizer" [Online]. Dosegljivo: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html [Dostopano 11. 5. 2016].

[68] scikit-learn developers. "sklearn.preprocessing.Imputer" [Online]. Dosegljivo: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html [Dostopano 11. 5. 2016]

[69] scikit-learn developers. "sklearn.svm.OneClassSVM" [Online]. Dosegljiv: http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html [Dostopano 19. 5. 2016].

[70] Z. Zhao, H. Liu, "Spectral feature selection for supervised and unsupervised learning", v zborniku *Proceedings of the 24th international conference on Machine learning*, Corvallis, Oregon, ZDA, jun. 2007, str. 1151—1157.

[71] Wikipedia. "Spectral graph theory" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Spectral_graph_theory [Dostopano 23. 6. 2016].

[72] Wikipedia. "Stop words" [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Stop_words [Dostopano 13. 5. 2016].

[73] P. Graham. "Stopwords for 50 languages in JSON format" [Online]. Dosegljivo: https://github.com/6/stopwords [Dostopano 13. 5. 2016].

[74] G. Gordon. "Support Vector Machines and Kernel Methods" [Online]. Dosegljiv: `https://www.cs.cmu.edu/~ggordon/SVMs/new-svms-and-kernels.pdf` [Dostopano 28. 6. 2016].

[75] L.J.P. van der Maaten. "t-Distributed Stochastic Neighbor Embedding Wins Merck Viz Challenge" [Online] Dosegljivo: `http://blog.kaggle.com/2012/11/02/t-distributed-stochastic-neighbor-embedding-wins-merck-viz-challenge/` [Dostopano 28. 6. 2016]

[76] L.J.P. van der Maaten. "t-SNE" [Online]. Dosegljivo: `http://lvdmaaten.github.io/tsne/` [Dostopano 27. 6. 2016].

[77] Wikipedia. "Term frequency–inverse document frequency" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Tf-idf` [Dostopano 4. 7. 2016].

[78] D. Kahneman, *Thinking, Fast and Slow*, New York: Farrar, Straus and Giroux, 2011.

[79] A. Zwitter Vitez, "Ugotavljanje avtorstva besedil: primer »Trenirkarjev«", v zborniku *Deveta konferenca JEZIKOVNE TEHNOLOGIJE Informacijska družba - IS 2014*, Ljubljana, Slovenija, okt. 2014, str. 131—134.

[80] Wikipedia. "Univerzalna decimalna klasifikacija" [Online]. Dosegljivo: `https://sl.wikipedia.org/wiki/Univerzalna_decimalna_klasifikacija` [Dostopano 26. 6. 2016].

[81] Wikipedia. "Variance" [Online]. Dosegljivo: `https://en.wikipedia.org/wiki/Variance` [Dostopano 26. 5. 2016].

[82] L.J.P. van der Maaten, G. Hinton, "Visualizing data using t-SNE", *Journal of Machine Learning Research*, nov. 2008, zv. 9, str. 2579—2605.

[83] TMRF IJCSA. "Web-based Recommender Systems, Year 2006: Volume III Issue III" [Online]. Dosegljivo: `http://www.tmrfindia.org/ijcsa/v33.html` [Dostopano 28. 6. 2016].

[84] S. Štajner, R. Evans, C. Orasan, R. Mitkov, "What can readability measures really tell us about text complexity", v zborniku *Proceedings of workshop on natural language processing for improving textual accessibility 2012*, Istambul, Turčija, maj 2012, str. 14—22.

[85] K. Beyer, J. Goldstein, R. Ramakrishnan, "When is "nearest neighbor" meaningful?", v zborniku *International conference on database theory*, Jeruzalem, Izrael, zv. 1540, jan. 1999, str. 217—235.

[86] M. Quijada. "Word complexity predictor" [Online]. Dosegljivo: `https://github.com/mauryquijada/word-complexity-predictor` [Dostopano 6. 6. 2016].

Priloge

Dodatek A

Podatki o uporabniku

Navajamo šifranta za izobrazbo in kategorijo uporabnikov v naši zbirki podatkov.

A.1 Stopnja izobrazbe uporabnika

0 ni podatka
1 7 razredov osnovne šole ali manj
2 dokončana osnovna šola
3 dokončana poklicna šola
4 dokončana srednja strokovna šola
5 dokončana gimnazija ali druga štiriletna srednja šola
6 dokončana višja šola
7 dokončana visoka šola, fakulteta ali umetn. akademija
8 magisterij
9 doktorat

A.2 Kategorije uporabnika

0 ni podatka
1 predšolski otroci
2 osnovnošolci
3 srednješolci
4 študenti (redni)

5 študenti (izredni, ob delu)
6 zaposleni
7 svobodni poklici
8 samostojni obrtniki/podjetniki
9 kmetje
10 gospodinje
11 upokojenci
12 nezaposleni
13 tuji državljani
14 zaposleni v matični ustanovi
15 zaposleni na univerzi
16 vojaki
17 častni člani
18 brezposelni
19 družinska izkaznica
20 podiplomski študenti
99 ostali
124 organizacijske enote ustanove
125 ravne osebe-zun. ustanove
126 note za medoddelčno izposojo

Dodatek B

Procesiranje naravnega jezika z knjižnico NLTK

NLTK je knjižnica za programski jezik python, namenjena procesiranju naravnega jezika (angl. natural language processing). Izbrali smo jo zaradi dobre dokumentacije, delne podpore v slovenščini in navidez enostavne tokenizacije, torej razdeljevanja besedil v podenote.

V nadaljevanju opišemo postopke, ki smo jih uporabljali pri obdelavi besedil.

B.1 Tokenizacija

Tokenizacija je postopek, kjer večje dele besedil razbijemo v manjše dele, torej poglavja v odstavke, odstavke v stavke in stavke v posamezne besede.

Razbitje poglavij v odstavke ni problematično. Glavni vir težav predstavlja razbitje odstavkov v stavke, ki je v knjižnici NLTK slabše realizirano. Posebej slabo se izkaže pri tokenizaciji zaporednih stavkov, ki imajo premi govor ali navedek, zato je bilo potrebno napisati popravljalnik, ki je vse stavke, ki so se končali z zaporedji črk ".«", "!«", "?«", "...«" ali "...«" na novo razdelil.

Razbitje stavkov v besede je problematično, ker je zgoraj našteta za-

poredja črk izločil kot eno "besedo". To je onemogočalo nadaljnje izračune premega govora ali navedkov. Težavo smo rešili z dodatnim popravljalnikom.

B.2 Oblikoslovno označevanje besedila

Med tokenizacijo stavkov v besede smo pognali tudi oblikoslovno označevanje besedila (angl. morphosyntactic description ali skrajšano MSD), ki je temeljitejša različica označevanja besednih vrst (angl. part-of-speech tagging, skraj. POS tagging) [18, 8], kjer posameznim besedam pripišemo oblikoslovne oznake.

Posamezni besedi sprva pripišemo besedno vrsto (samostalnik, glagol, pridevnik, prislov, ...), potem pa jo opišemo še podrobneje. Kot primer, samostalnike razvrstimo tudi v lastna in obča imena, glagolom določimo vid, obliko, osebo in spol, pridevnikom vrsto, stopnjo, spol, število in tako dalje. Vsaka opisna kombinacija dobi svoj niz oznak (t.i. oznaka MSD), ki so definirane v priporočilih oblikoskladenjske specifikacije JOS [8].

Primer stavka, kjer so besede označene z oznakami MSD, prikazuje tabela B.1.

Tabela B.1: Oblikoslovno označevanja stavka v korpusu ccGigafida.

besede stavka	To	bi	bralcem	priskutilo	zajtrk	.
lema	to	biti	bralec	priskutiti	zajtrk	.
MSD oznaka	Zk-sei	Gp-g	Sommd	Ggdd-es	Sometn	/
interpretacija MSD oznake	zaimek vrsta=kazalni spol=srednji število=ednina sklon=imenovalnik	glagol vrsta=pomožni oblika=pogojnik	samostalnik vrsta=občno ime spol=moški število=množina sklon=dajalnik	glagol vrsta=glavni vid=dovršni oblika=deležnik število=ednina spol=srednji	samostalnik vrsta=občno ime spol=moški število=ednina sklon=tožilnik živost=ne	

Primer zapisa istega stavka v korpusu ccGigafida (del datoteke F0016336.xml):

```
<s>
   <w msd="Zk-sei" lemma="ta">To</w>
   <S/>
   <w msd="Gp-g" lemma="biti">bi</w>
   <S/>
   <w msd="Sommd" lemma="bralec">bralcem</w>
   <S/>
   <w msd="Ggdd-es" lemma="priskutiti">priskutilo</w>
   <S/>
   <w msd="Sometn" lemma="zajtrk">zajtrk</w>
   <c>.</c>
   <S/>
</s>
```

Korpus

Korpus je elektronska zbirka avtentičnih besedil. Besedila so izbrana po vnaprej določenih merilih in z določenim ciljem. Opremljena so z dodanimi jezikovnimi podatki.

Korpus, kjer so besede že označene z oznakami MSD v našem primeru potrebujemo zato, da lahko algoritem za strojno učenje naučimo (glej B.2), kako pripisovati novim besedam oznake MSD.

Za učno in testno množico smo izbrali javno dostopen korpus slovenskih besedil ccGigafida [60], ki obsega učno množico 100 milijonov besed najrazličnejših zvrsti (leposlovje, učbeniki, dnevni časopisi, revije, internetni zapisi). Je podmnožica korpusa Gigafida [60], ki obsega kar 1,2 miljarde besed. Besedila v korpusu ccGigafida so v formatu XML in imajo za vsako besedo definirano lemo ter MSD oznako po oblikoskladenjski specifikaciji JOS [8].

Postopek označevanja V našem primeru želimo s strojnim učenjem opraviti oblikoslovno označevanje. Učenje izvedemo na korpusu. Nato poženemo klasifikacijo in vsaki besedi pripišemo MSD kodo. To je tudi končen rezultat strojnega učenja.

Poizkusili smo uporabiti tudi prosto dostopen označevalnik za slovenski jezik Obeliks [45], ki pa nam ga zaradi omejitev programske opreme ni uspelo usposobiti za delovanje.

Strojno učenje Strojno učenje je definirano kot pridobivanje znanja na podlagi učnih podatkov. Posledično lahko odgovorimo na vprašanja o podatkih, ki niso bila v učnih podatkih.

Tipe strojnega učenja razdelimo glede na posredovano informacijo:

- **Nadzorovano učenje** (angl. supervised learning) – algoritem dobi učno množico z vhodnimi podatki in želenimi izhodnimi vrednostmi, na katerih se nauči klasifikacije novih, še nepoznanih vhodnih podatkov. Želene izhodne vrednosti ponavadi določa človek.

- **Nenadzorovano učenje** (angl. unsupervised learning) – algoritem razdeli vhodne podatke v več gruč v postopku, imenovanem gručenje.

- **Vzpodbujevalno učenje** (angl. reinforcement learning) – algoritem se uči z nagrajevanjem in kaznovanjem.

V primeru označevanja MSD gre za **nadzorovano strojno učenje**, saj imamo označeno učno množico. Iz nje lahko zaradi računskih omejitev izločimo dovolj veliko učno podmnožico in jo uporabimo za učenje. Pri tem moramo poskrbeti za čimboljše ohranjanje deležev posameznih kategorij besedil. Treba se je zavedati, da korpus ccGigafida ni brez napak, saj je bil tudi sam označen z označevalnikom Obeliks [45].

Ker smo obdelali večje število knjig, je bilo treba poskrbeti za hitrost označevanja in pri tem doseči čimvečjo natačnost. Čas učenja označevalnika ni bil pomemben, saj je bistveno manjši od označevanja vseh knjig.

Na naši strojni in programski opremi je povprečen čas obdelave ene knjige trajal 93,7 sekunde. Pri 1160 knjigah je bil skupen čas trajanja obdelave 1 dan, 6 ur ter 17 minut. Učenje izbranega klasifikatorja traja 59 minut in 28 sekund.

Knjižnica scikit-learn Testirali smo s pomočjo sledečih označevalnikov, ki so podprti v knjižnici *scikit-learn* [62] (uporablja se tudi okrajšava *sklearn*) za programski jezik *python* in se lepo povežejo z knjižnico *NLTK*.

Posebnost je označevalnik MegaM [40], ki deluje kot zunanja knjižnica. Na operacijskem sistemu OSX ga je težje vkomponirati v *python* in *NLTK*. Seznam označevalnikov, na katerih smo testirali označevanje, je sledeč:

- N-gram,
- Naivni Bayes,
- Odločitvena drevesa (angl. decision tree),
- Model maksimalne entropije (točneje MEGA Model ali skr. MegaM),
- Brillov označevalnik.

Vse metode smo testirali z različnimi velikostmi naključno izbrane učne podmnožice korpusa ccGigafida. Testirali smo tudi z različnimi nastavitvami algoritmov. Glede na pretekle izsledke smo se trudili iterativno izboljševati hitrost in natančnost.

Zaradi želenih lastnosti, torej dobre natančnosti in hitrosti, smo na koncu izbrali označevalnik Brill. Ta deluje dvofazno, v prvi fazi pa potrebuje dodaten označevalnik. Za to nalogo smo izbrali najhitrejšega, torej n-gramski označevalnik nivoja n = 3 (trigram). V drugi fazi se Brillov označevalnik nauči popravljati oznake prvega označevalnika. Nauči se torej novih pravil, ki povečajo natančnost končnega označevanja.

B.3 Lematizacija

Krnjenje (alg. stemming) je postopek, pri katerem odrežemo končnico besede in dobimo krn, ki ni nujno obstoječa knjižna beseda. Pri tem prihaja do izgube informacij, saj ima več besed isti koren, prav tako pa ni upoštevan kontekst, v katerem se beseda nahaja.

Za razliko od krnjenja pri lematizaciji besedi oz. besedni enoti določimo osnovno slovarsko obliko, imenovano lema. Pri tem upoštevamo tudi kontekst, v katerem se beseda nahaja. Tako kot krnjenje postopek opravljamo ročno ali računalniško. V slednjem primeru je lematizacija zahtevnejši postopek, saj zahteva predhodno oblikoslovno označevanje, ponavadi pa naredimo

oba postopka zaporedno. Lematizacijo uporabimo, ko nas zanima točnejši pomen besed.

Posebnost slovenščine je potencialno precejšnja razlika med krnom in lemo, kar za angleščino ne velja. Besedi "boljši" pripada lema "dober". Krn besede "boljši" je "bolj".

Uporabili smo lematizator Lemmagen [38], ki ga je bilo za uporabo v operacijskem sistemu OSX in jeziku python razl. 3 ustrezno prilagoditi.

Dodatek C

Optimizacija shranjevanja

Vnos dokumentov (knjig, uporabnikov) v MongoDB zahteva čas. Zaradi obilice vnosov (glej tabelo C.1) je bilo treba vnašanje hitrostno optimizirati. Zato je namesto vsakokratnega posameznega vnašanja s t. i. funkcijo vstavljanja bolje inicializirati sveženj operacij (angl. bulk operation), v katerega dodajamo operacije in jih poženemo na koncu. Tabela C.2 prikazuje primerjavo shranjevanja brez in z optimizacijo z razkosanjem poglavij v odstavke, stavke in besede za eno knjigo. Pohitritev je za konkreten primer kar 2,65-kratna, kar nam pri veliko vnosih prihrani precej časa.

Tabela C.1: Primer velikosti posameznih zbirk (angl. collections).

ime zbirke (collection)	**namen**	**št. dokumentov**	**velikost**
books	hranjenje knjig, njihovih vektorjev značilk in gručenj	1.147	4,7 MB
chapters	hranjenje poglavij	34.291	368,6 MB
paragraphs	hranjenje odstavkov	206.517	974,1 MB
sentences	hranjenje stavkov	4.496.481	4,1 GB
words	hranjenje besed	76.111.982	17,8 GB
users	hranjenje uporabnikov, njihovih vektorjev značilk in gručenj	10.813	4,2 MB
word_frequency	hranjenje vseh različnih besed iz korpusa za izračunavanje pogostosti	1.724.308	137,6 MB
lemma_frequency_aoa_conc	hranjenje vseh lem iz korpusa za izračunavanje pogostosti, leta pridobitve in konkretnosti	1.014.528	63,3 MB

Tabela C.2: Tabela primerjave shranjevanja brez in z optimizacijo.

Brez optimizacije (običajno shranjevanje)	13,506 s
Z optimizacijo (shranjevanje s svežnjem oper.)	5,083 s

Primer kode v pythonu za shranjevanje enega dokumenta:

```
def insert_chapter_db(db, mongo_book_id, chapter_json, order_in_book):
    chapter_json["book_id"] = mongo_book_id
    chapter_json["order_in_book"] = order_in_book
    result = db[CHAPTERS_COLLECTION].insert_one(chapter_json, {"$currentDate": {"last_modified": True
        }})
    chapter_mongo_id = result.inserted_id
    return chapter_mongo_id
```

Primer kode v pythonu za shranjevanje s svežnjem operacij:

```
from lib.mongo_connection import get_db
from lib.frequency import lemma_frequency_db
from lib.concreteness_and_aoa import get_aoa_db, get_concreteness_db

db = get_db()
all_lemmas = db["lemma_frequency"].find({})

bulk = db["lemma_frequency_aoa_conc"].initialize_unordered_bulk_op()

for lemma in all_lemmas:
   lemma_data = {}
   lemma_data["lemma"] = lemma["word"]
   lemma_data["freq"] = lemma_frequency_db(db, lemma["word"])
   lemma_data["conc"] = get_concreteness_db(db, lemma["word"])
   lemma_data["aoa"] = get_aoa_db(db, lemma["word"])
   bulk.insert(lemma_data)

bulk.execute()
db["lemma_frequency_aoa_conc"].create_index({ "lemma": 1 }, { unique: true })
```